VISIT…

ELSEVIER FINANCE

Computational Finance

Using C and C#

George Levy

AP

Computational Finance Using C and C#

Quantitative Finance Series

Aims and Objectives

- Books based on the work of financial market practitioners and academics
- Presenting cutting-edge research to the professional/practitioner market
- Combining intellectual rigour and practical application
- Covering the interaction between mathematical theory and financial practice
- To improve portfolio performance, risk management and trading book performance
- Covering quantitative techniques

Market

Brokers/Traders; Actuaries; Consultants; Asset Managers; Fund Managers; Regulators; Central Bankers; Treasury Officials; Technical Analysis; and Academics for Masters in Finance and MBA market.

Series Titles

Computational Finance Using C and C#
The Analytics of Risk Model Validation
Forecasting Expected Returns in the Financial Markets
Corporate Governance and Regulatory Impact on Mergers and Acquisitions
International Mergers and Acquisitions Activity Since 1990
Forecasting Volatility in the Financial Markets, Third Edition
Venture Capital in Europe
Funds of Hedge Funds
Initial Public Offerings
Linear Factor Models in Finance
Computational Finance
Advances in Portfolio Construction and Implementation
Advanced Trading Rules, Second Edition
Real R&D Options
Performance Measurement in Finance
Economics for Financial Markets
Managing Downside Risk in Financial Markets
Derivative Instruments: Theory, Valuation, Analysis
Return Distributions in Finance

Series Editor: Dr Stephen Satchell

Dr Satchell is Reader in Financial Econometrics at Trinity College, Cambridge; Visiting Professor at Birkbeck College, City University Business School and University of Technology, Sydney. He also works in a consultative capacity to many firms, and edits the journal *Derivatives: use, trading and regulations* and the *Journal of Asset Management*.

Computational Finance Using C and C#

George Levy

AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK
OXFORD • PARIS • SAN DIEGO • SAN FRANCISCO • SINGAPORE
SYDNEY • TOKYO

Academic Press is an imprint of Elsevier

Cover image courtesy of iStockphoto
Academic Press is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
84 Theobald's Road, London WC1X 8RR, UK

Copyright © 2008, Elsevier Ltd. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, E-mail: permissions@elsevier.com. You may also complete your request on-line via the Elsevier homepage (http://elsevier.com), by selecting "Support & Contact" then "Copyright and Permission" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data

Levy, George.
Computational Finance Using C and C# / George Levy.
p. cm. – (Quantitative finance)
Includes bibliographical references and index.
ISBN-13: 978-0-7506-6919-1 (alk. paper) 1. Finance-Mathematical models. I. Title.
HG106.L484 2008
332.0285'5133-dc22
2008000470

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

For information on all Academic Press publications
visit our Web site at www.books.elsevier.com

Printed in the United States of America
08 09 10 11 9 8 7 6 5 4 3 2 1

Working together to grow
libraries in developing countries
www.elsevier.com | www.bookaid.org | www.sabre.org
ELSEVIER BOOK AID International Sabre Foundation

To my parents Paul and Paula

This page intentionally left blank

Contents

Preface xi

1 Overview of financial derivatives 1

2 Introduction to stochastic processes 5
2.1 Brownian motion 5
2.2 A Brownian model of asset price movements 9
2.3 Ito's formula (or lemma) 10
2.4 Girsanov's theorem 12
2.5 Ito's lemma for multiasset geometric Brownian motion 13
2.6 Ito product and quotient rules in two dimensions 15
2.7 Ito product in n dimensions 18
2.8 The Brownian bridge 19
2.9 Time-transformed Brownian motion 21
2.10 Ornstein–Uhlenbeck process 24
2.11 The Ornstein–Uhlenbeck bridge 27
2.12 Other useful results 31
2.13 Selected problems 33

3 Generation of random variates 37
3.1 Introduction 37
3.2 Pseudo-random and quasi-random sequences 38
3.3 Generation of multivariate distributions: independent variates 41
3.4 Generation of multivariate distributions: correlated variates 47

4 European options 59
4.1 Introduction 59
4.2 Pricing derivatives using a martingale measure 59
4.3 Put call parity 60
4.4 Vanilla options and the Black–Scholes model 62
4.5 Barrier options 85

5 Single asset American options 97
5.1 Introduction 97
5.2 Approximations for vanilla American options 97
5.3 Lattice methods for vanilla options 114

5.4 Grid methods for vanilla options 135
5.5 Pricing American options using a stochastic lattice 172

6 Multiasset options 181
6.1 Introduction 181
6.2 The multiasset Black–Scholes equation 181
6.3 Multidimensional Monte Carlo methods 183
6.4 Introduction to multidimensional lattice methods 185
6.5 Two asset options 190
6.6 Three asset options 201
6.7 Four asset options 205

7 Other financial derivatives 209
7.1 Introduction 209
7.2 Interest rate derivatives 209
7.3 Foreign exchange derivatives 228
7.4 Credit derivatives 232
7.5 Equity derivatives 237

8 C# portfolio pricing application 245
8.1 Introduction 245
8.2 Storing and retrieving the market data 254
8.3 The `PricingUtils` class and the `Analytics_MathLib` 262
8.4 Equity deal classes 267
8.5 FX deal classes 280

Appendix A: The Greeks for vanilla European options 289
A.1 Introduction 289
A.2 Gamma 290
A.3 Delta 291
A.4 Theta 292
A.5 Rho 293
A.6 Vega 294

Appendix B: Barrier option integrals 295
B.1 The down and out call 295
B.2 The up and out call 298

Appendix C: Standard statistical results 303
C.1 The law of large numbers 303
C.2 The central limit theorem 303
C.3 The variance and covariance of random variables 305
C.4 Conditional mean and covariance of normal distributions 310
C.5 Moment generating functions 311

Appendix D: Statistical distribution functions 313
D.1 The normal (Gaussian) distribution 313
D.2 The lognormal distribution 315
D.3 The Student's *t* distribution 317
D.4 The general error distribution 319

Appendix E: Mathematical reference 321
E.1 Standard integrals 321
E.2 Gamma function 321
E.3 The cumulative normal distribution function 322
E.4 Arithmetic and geometric progressions 323

Appendix F: Black–Scholes finite-difference schemes 325
F.1 The general case 325
F.2 The log transformation and a uniform grid 325

Appendix G: The Brownian bridge: alternative derivation 329

Appendix H: Brownian motion: more results 333
H.1 Some results concerning Brownian motion 333
H.2 Proof of Eq. (H.1.2) 334
H.3 Proof of Eq. (H.1.4) 335
H.4 Proof of Eq. (H.1.5) 335
H.5 Proof of Eq. (H.1.6) 335
H.6 Proof of Eq. (H.1.7) 338
H.7 Proof of Eq. (H.1.8) 338
H.8 Proof of Eq. (H.1.9) 338
H.9 Proof of Eq. (H.1.10) 339

Appendix I: The Feynman–Kac formula 341

Appendix J: Answers to problems 343
J.1 Problem 1 343
J.2 Problem 2 344
J.3 Problem 3 345
J.4 Problem 4 346
J.5 Problem 5 346
J.6 Problem 6 347
J.7 Problem 7 348
J.8 Problem 8 350
J.9 Problem 9 350
J.10 Problem 10 352
J.11 Problem 11 354

References 355

Index 361

This page intentionally left blank

Preface

This book builds on the author's previous book *Computational Finance: Numerical Methods for Pricing Financial Instruments*, which contained information on pricing equity options using C code. The current book covers the following instrument types:

- Equity derivatives
- Interest rate derivatives
- Foreign exchange derivatives
- Credit derivatives

There is also an extensive final chapter which demonstrates how a C-based analytics pricing library can be used by C# portfolio valuation software. In addition this application:

- illustrates the use of C# dictionaries, abstract classes and .NET InteropServices
- permits the reader to value bespoke portfolios
- allows market data to be specified via a configuration file
- contains a generic basket pricer for which the reader can specify the payoff function
- can be freely downloaded for use by the reader.

The current book also contains increased coverage of stochastic processes, Ito calculus and Monte Carlo simulation. These topics are supported by practical applications and solved example problems.

In addition the Numerical Algorithms Group (NAG) have allowed readers to enjoy an extended trial licence for the NAG C library and associated financial routines from the following url: www.nag.co.uk/market/elsevier_glevy. The NAG C library may be called into C# and provides a large suite of mathematical routines addressing many areas covered in this book (random numbers, statistical distributions, option pricing, correlation and covariance matrices etc.).

Computational Finance Using C and C# also includes supporting software that may be downloaded for free. The software consists of executable files, configuration files and results files. With these files the user can run the example portfolio application in Chapter 8 and change the portfolio composition and the attributes of the deals.

Additional upgrade software is available for purchase with *Computational Finance Using C and C#*. The software includes:

- Code to run all the C, C# and Excel examples in the book

- Complete C source code for the Analytics_Mathlib math library that is used in the book
- C# source code, market data and portfolio files for the portfolio application described in Chapter 8

All the C/C# software in the book can be compiled using either Visual Studio .NET 2005, or the freely available Microsoft Visual C#/C++ Express Editions.

I would like to take this opportunity of thanking my wife Kathy for her support.

In addition I am grateful to Karen Maloney of Elsevier for her patience with regard to the book's delivery date, and Dr. Stephen Satchell of Trinity College Cambridge for allowing me the opportunity to write a sequel.

George Levy
Benson, Oxfordshire, UK
2008

1 Overview of financial derivatives

A financial derivative is a contract between two counterparties (here referred to as *A* and *B*) which *derives* its value from the *state* of underlying financial quantities. We can further divide derivatives into those that carry a future *obligation* and those that don't. In the financial world a derivative which gives the owner the *right* but not the *obligation* to participate in a given financial contract is called an *option*. We will now illustrate this using both a Foreign Exchange Forward contract and a Foreign Exchange option.

Foreign Exchange Forward—a contract with an obligation

In a Foreign Exchange Forward contract a certain amount of foreign currency will be bought (or sold) at a future date using a prearranged foreign exchange rate.

For instance, counterparty *A* may own a Foreign Exchange Forward which, in one year's time, contractually obliges *A* to purchase from *B* the sum of $200 for £100. At the end of one year several things may have happened.

(i) The value of the pound may have decreased with respect to the dollar
(ii) The value of the pound may have increased with respect to the dollar
(iii) Counterparty *B* may refuse to honor the contract—*B* may have gone bust, etc.
(iv) Counterparty *A* may refuse to honor the contract—*A* may have gone bust, etc.

We will now consider events (i)–(iv) from *A*'s perspective.

Firstly, if (i) occurs then *A* will be able to obtain $200 for less than the current market rate, say £120. In this case the $200 can be bought for £100 and then immediately sold for £120, giving a profit of £20. However, this profit can only be realized if *B* honors the contract—that is, event (iii) does not happen.

Secondly, when (ii) occurs then *A is obliged to* purchase $200 for *more* than the current market rate, say £90. In this case the $200 are bought for £100 but could have been bought for only £90, giving a loss of £10.

The probability of events (iii) and (iv) occurring are related to the *Credit Risk* associated with counterparty *B*. The value of the contract to *A* is not affected by (iv), although *A* may be sued if both (ii) and (iv) occur. Counterparty *A* should only be concerned with the possibility of events (i) and (iii) occurring—that is, the probability that the contract is worth a positive amount in one year

and the probability that *B* will honor the contract (which is one minus the probability that event (iii) will happen).

From *B*'s point of view the important Credit Risk is when both (ii) and (iv) occur—that is, when the contract has positive value but counterparty *A* defaults.

Foreign Exchange option—a contract without an obligation

A Foreign Exchange option is similar to the Foreign Exchange Forward, the difference being that if event (ii) occurs then *A* is not *obliged* to buy dollars at an unfavorable exchange rate. To have this flexibility *A* needs to *buy* a Foreign Exchange *option* from *B*, which here can be regarded as insurance against unexpected exchange rate fluctuations.

For instance, counterparty *A* may own a Foreign Exchange option which, in one year, contractually *allows A* to purchase from *B* the sum of $200 for £100. As before, at the end of one year the following may have happened:

(i) The value of the pound may have decreased with respect to the dollar
(ii) The value of the pound may have increased with respect to the dollar
(iii) Counterparty *B* may refuse to honor the contract—*B* may have gone bust, etc.
(iv) Counterparty *A* may have gone bust, etc.

We will now consider events (i)–(iv) from *A*'s perspective.

Firstly, if (i) occurs then *A* will be able to obtain $200 for less than the current market rate, say £120. In this case the $200 can be bought for £100 and then immediately sold for £120, giving a profit of £20. However, this profit can only be realized if *B* honors the contract—that is, event (iii) does not happen.

Secondly, when (ii) occurs then *A* will decide not to purchase $200 for *more* than the current market rate; in this case the option is worthless.

We can thus see that *A* is still concerned with the *Credit Risk* when events (i) and (iii) occur simultaneously.

The Credit Risk from counterparty *B*'s point of view is different. *B* has sold to *A* a Foreign Exchange option, which matures in one year, and has already received the money—the current fair price for the option. Counterparty *B* has no Credit Risk associated with *A*. This is because if event (iv) occurs, and *A* goes bust, it doesn't matter to *B* since the money for the option has already been received. On the other hand, if event (iii) occurs *B* may be sued by *A* but *B* still has no Credit Risk associated with *A*.

This book considers the valuation of financial derivatives that carry obligations and also financial options.

Chapters 1–7 deal with both the theory of stochastic processes and the pricing of financial instruments. In Chapter 8 this information is then applied to a C# portfolio valuer. The application is easy to use (the portfolios and current market rates are defined in text files) and can also be extended to include new trade types.

The book has been written so that (as far as possible) financial mathematics results are derived from first principles.

Finally, the appendices contain various information, which we hope the reader will find useful.

This page intentionally left blank

2 Introduction to stochastic processes

2.1 Brownian motion

Brownian motion is named after the botanist Robert Brown who used a microscope to study the fertilization mechanism of flowering plants. He first observed the random motion of pollen particles (obtained from the American species *Clarkia pulchella*) suspended in water, and wrote:

> The fovilla or granules fill the whole orbicular disk but do not extend to the projecting angles. They are not sphaerical but oblong or nearly cylindrical, and the particles have manifest motion. This motion is only visible to my lens which magnifies 370 times. The motion is obscure yet certain . . .
>
> Robert Brown, 12th June 1827; see Ramsbottom (1932)

It appears that Brown considered this motion no more than a curiosity (he believed that the particles were *alive*) and continued *undistracted* with his botanical research. The full significance of his observations only became apparent about eighty years later when it was shown (Einstein, 1905) that the motion is caused by the collisions that occur between the pollen grains and the water molecules. In 1908 Perrin (1909) was finally able to confirm Einstein's predictions experimentally. His work was made possible by the development of the ultramicroscope by Richard Zsigmondy and Henry Siedentopf in 1903. He was able to work out from his experimental results and Einstein's formula the size of the water molecule and a precise value for Avogadro's number. His work established the physical theory of Brownian motion and ended the skepticism about the existence of atoms and molecules as actual physical entities. Many of the fundamental properties of Brownian motion were discovered by Paul Levy (Levy, 1939, 1948), and the first mathematically rigorous treatment was provided by Norbert Wiener (Wiener, 1923, 1924). Karatzas and Shreve (1991) is an excellent textbook on the theoretical properties of Brownian motion, while Shreve, Chalasani, and Jha (1997) provides much useful information concerning the use of Brownian processes within finance.

Brownian motion is also called a *random walk*, a Wiener process, or sometimes (more poetically) the *drunkard's walk*. We will now present the three fundamental properties of Brownian motion.

2.1.1 The properties of Brownian motion

In formal terms a process $W = (W_t : t \geqslant 0)$ is (one-dimensional) Brownian motion if:

(i) W_t is continuous, and $W_0 = 0$,
(ii) $W_t \sim \mathrm{N}(0, t)$,
(iii) The increment $\mathrm{d}W_t = W_{t+\mathrm{d}t} - W_t$ is normally distributed as $\mathrm{d}W_t \sim \mathrm{N}(0, \mathrm{d}t)$, so $E[\mathrm{d}W_t] = 0$ and $\mathrm{Var}[\mathrm{d}W_t] = \mathrm{d}t$. The increment $\mathrm{d}W_t$ is also independent of the history of the process up to time t.

From (iii) we can further state that, since the increments $\mathrm{d}W_t$ are independent of past values W_t, a Brownian process is also a *Markov* process. In addition we shall now show that a Brownian process is also a *martingale* process.

In a martingale process $P_t, t \geqslant 0$, the conditional expectation $E[P_{t+\mathrm{d}t}|\mathcal{F}_t] = P_t$, where $\mathcal{F}_t$ is called the *filtration* generated by the process and contains the information learned by observing the process up to time t. Since for Brownian motion we have

$$\begin{aligned} E[W_{t+\mathrm{d}t}|\mathcal{F}_t] &= E\big[(W_{t+\mathrm{d}t} - W_t) + W_t|\mathcal{F}_t\big] = E[W_{t+\mathrm{d}t} - W_t] + W_t \\ &= E[\mathrm{d}W_t] + W_t = W_t \end{aligned}$$

where we have used the fact that $E[\mathrm{d}W_t] = 0$. Since $E[W_{t+\mathrm{d}t}|\mathcal{F}_t] = W_t$ the Brownian motion W is a martingale process.

Using property (iii) we can also derive an expression for the covariance of Brownian motion. The independent increment requirement means that for the n times $0 \leqslant t_0 < t_1 < t_2 < \cdots < t_n < \infty$ the random variables $W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}}$ are independent. So

$$\mathrm{Cov}[W_{t_i} - W_{t_{i-1}}, W_{t_j} - W_{t_{j-1}}] = 0, \quad i \neq j \tag{2.1.1}$$

We will show that $\mathrm{Cov}[W_s, W_t] = s \wedge t$.

Proof. Using $W_{t_0} = 0$, and assuming $t \geqslant s$ we have

$$\mathrm{Cov}[W_s - W_{t_0}, W_t - W_{t_0}] = \mathrm{Cov}[W_s, W_t] = \mathrm{Cov}\big[W_s, W_s + (W_t - W_s)\big]$$

From Appendix C.3.2 we have

$$\begin{aligned} \mathrm{Cov}\big[W_s, W_s + (W_t - W_s)\big] &= \mathrm{Cov}[W_s, W_s] + \mathrm{Cov}[W_s, W_t - W_s] \\ &= \mathrm{Var}[W_s] + \mathrm{Cov}[W_s, W_t - W_s] \end{aligned}$$

Therefore

$$\mathrm{Cov}[W_s, W_t] = s + \mathrm{Cov}[W_s, W_t - W_s]$$

Now

$$\mathrm{Cov}[W_s, W_t - W_s] = \mathrm{Cov}[W_s - W_{t_0}, W_t - W_s] = 0$$

where we have used Eq. (2.1.1) with $n = 2, t_1 = s$ and $t_2 = t$.

We thus obtain

$$\mathrm{Cov}[W_s, W_t] = s$$

So

$$\mathrm{Cov}[W_s, W_t] = s \wedge t \tag{2.1.2}$$

□

We will now consider the Brownian increments over the time interval $\mathrm{d}t$ in more detail. Let us first define the process X such that:

$$\mathrm{d}X_t = \mathrm{d}W_t \tag{2.1.3}$$

where $\mathrm{d}W_t$ is a random variable drawn from a normal distribution with mean zero and variance $\mathrm{d}t$, which we denote as $\mathrm{d}W_t \sim \mathrm{N}(0, \mathrm{d}t)$. Equation (2.1.3) can also be written in the equivalent form:

$$\mathrm{d}X_t = \sqrt{\mathrm{d}t}\,\mathrm{d}Z \tag{2.1.4}$$

where $\mathrm{d}Z$ is a random variable drawn from a *standard* normal distribution (that is a normal distribution with zero mean and unit variance).

Equations (2.1.3) and (2.1.4) give the incremental change in the value of X over the time interval $\mathrm{d}t$ for *standard* Brownian motion.

We shall now generalize these equations slightly by introducing the extra (*volatility*) parameter σ which controls the variance of the process. We now have:

$$\mathrm{d}X_t = \sigma\,\mathrm{d}W_t \tag{2.1.5}$$

where $\mathrm{d}W_t \sim \mathrm{N}(0, \mathrm{d}t)$ and $\mathrm{d}X_t \sim \mathrm{N}(0, \sigma^2\,\mathrm{d}t)$. Equation (2.1.5) can also be written in the equivalent form:

$$\mathrm{d}X_t = \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z, \quad \mathrm{d}Z \sim \mathrm{N}(0, 1) \tag{2.1.6}$$

or equivalently

$$\mathrm{d}X_t = \sqrt{\mathrm{d}t}\,\mathrm{d}\widehat{Z}, \quad d\widehat{Z} \sim \mathrm{N}\left(0, \sigma^2\right) \tag{2.1.7}$$

We are now in a position to provide a mathematical description of the movement of the pollen grains observed by Robert Brown in 1827. We will start by assuming that the container of water is perfectly level. This will ensure that there is no drift of the pollen grains in any particular direction. Let us denote the position of a particular pollen grain at time t by X_t, and set the position at $t = 0$, X_{t_0}, to zero. The statistical distribution of the grain's position, X_T, at some later time $t = T$, can be found as follows:

Let us divide the time T into n equal intervals $\mathrm{d}t = T/n$. Since the position of the particle changes by the amount $\mathrm{d}X_i = \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i$ over the ith time interval $\mathrm{d}t$, the final position X_T is given by:

$$X_T = \sum_{i=1}^{n}\left(\sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i\right) = \sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i$$

Since $\mathrm{d}Z_i \sim \mathrm{N}(0, 1)$, by the Law of Large Numbers (see Appendix C.1), we have that the expected value of position X_T is:

$$E[X_T] = \sigma\sqrt{\mathrm{d}t}\,E\left[\sum_{i=1}^{n}\mathrm{d}Z_i\right] = 0$$

The variance of the position X_T is:

$$\mathrm{Var}[X_T] = \mathrm{Var}\left[\sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i\right] = \sigma^2\,\mathrm{d}t\,\mathrm{Var}\left[\sum_{i=1}^{n}\mathrm{d}Z_i\right] \tag{2.1.8}$$

Since all the $\mathrm{d}Z_i$ variates are IID $\mathrm{N}(0, 1)$ we have $\mathrm{Var}[\mathrm{d}Z_i] = 1$ and $\mathrm{Var}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n}\mathrm{Var}[X_i]$ (see Appendix C.3.1).

Thus

$$\mathrm{Var}[X_T] = \sigma^2\,\mathrm{d}t\sum_{i=1}^{n}\mathrm{Var}[\mathrm{d}Z_i] = \sigma^2\,\mathrm{d}t\sum_{i=1}^{n}1 \tag{2.1.9}$$

which gives:

$$\mathrm{Var}[X_T] = \sigma^2 n\,\mathrm{d}t = T\sigma^2 \tag{2.1.10}$$

So, at time T, the position of the pollen grain X_T is distributed as $X_T \sim \mathrm{N}(0, T\sigma^2)$.

If the water container is not perfectly level then the pollen grains will exhibit drift in a particular direction. We can modify Eq. (2.1.5) to take this into account as follows:

$$\mathrm{d}X_t = \mu\,\mathrm{d}t + \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i, \quad \mathrm{d}Z_i \sim \mathrm{N}(0, 1), \tag{2.1.11}$$

or equivalently

$$\mathrm{d}X_t = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t, \quad \mathrm{d}W_t \sim \mathrm{N}(0, \mathrm{d}t), \tag{2.1.12}$$

where we have included the *constant* drift μ. Proceeding in a similar manner to that for the case of *zero drift* Brownian motion we have:

$$\begin{aligned} X_T &= \sum_{i=1}^{n}\left(\mu\,\mathrm{d}t + \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i\right) = \mu\sum_{i=1}^{n}\mathrm{d}t + \sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i \\ &= \mu T + \sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i \end{aligned}$$

which gives

$$E[X_T] = E\left[\mu T + \sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i\right]$$

$$E[X_T] = \mu T + \sigma\sqrt{\mathrm{d}t}\,E\left[\sum_{i=1}^{n}\mathrm{d}Z_i\right] = \mu T$$

The variance of the position X_T is:

$$\mathrm{Var}[X_T] = \mathrm{Var}\left[\mu T + \sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i\right] = \mathrm{Var}\left[\sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i\right]$$

Here we have used the fact (see Appendix C.3.1) that $\mathrm{Var}[a + bX] = b^2\,\mathrm{Var}[X]$, where $a = \mu T$, and $b = 1$. From Eqs. (2.1.9) and (2.1.10) we have:

$$\mathrm{Var}[X_T] = \mathrm{Var}\left[\sigma\sqrt{\mathrm{d}t}\sum_{i=1}^{n}\mathrm{d}Z_i\right] = T\sigma^2$$

So, at time T, the position of the pollen grain X_T is distributed as $X_T \sim \mathrm{N}(\mu T, T\sigma^2)$.

We have just shown that when we vary the drift of a Brownian motion, its volatility remains unchanged. This is a very important property and (as we will see later) is used extensively in the theory of derivative pricing.

2.2 A Brownian model of asset price movements

In the previous section we showed how Brownian motion can be used to describe the random motion of small particles suspended in a liquid. The first attempt at using Brownian motion to describe financial asset price movements was provided by Bachelier (1900). This, however, only had limited success because the *significance* of a given *absolute* change in asset price depends on the original asset price. For example, a £1 increase in the value of a share originally worth £1.10 is much more *significant* than a £1 increase in the value of a share originally worth £100. It is for this reason that asset price movements are generally described in terms of *relative* or percentage changes. For example, if the £1.10 share increases in value by 11 pence and the £100 share increases in value by £10, then both of these price changes have the same significance, and correspond to a 10 percent increase in value. The idea of relative price changes in the value of a share can be formalized by defining a quantity called the *return*, R_t, of a share at time t. The return R_t is defined as follows:

$$R_t = \frac{S_{t+\mathrm{d}t} - S_t}{S_t} = \frac{\mathrm{d}S_t}{S_t} \tag{2.2.1}$$

where $S_{t+\mathrm{d}t}$ is the value of the share at time $t + \mathrm{d}t$, S_t is the value of the share at time t, and $\mathrm{d}S_t$ is the change in value of the share over the time interval $\mathrm{d}t$. The percentage return R^* over the time interval $\mathrm{d}t$ is simply defined as $R^* = 100 \times R_t$.

We are now in a position to construct a simple Brownian model of asset price movements; further information on Brownian motion within finance can be found in Shreve, Chalasani, and Jha (1997).

The asset *return* at time t is now given by:

$$R_t = \frac{\mathrm{d}S_t}{S_t} = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t, \quad \mathrm{d}W_t \sim \mathrm{N}(0, \mathrm{d}t), \tag{2.2.2}$$

or equivalently:

$$\mathrm{d}S_t = S_t\mu\,\mathrm{d}t + S_t\sigma\,\mathrm{d}W_t \tag{2.2.3}$$

The process in Eqs. (2.2.2) and (2.2.3) is termed *geometric Brownian motion*; which we will abbreviate as GBM. This is because the relative (rather than absolute) price changes follow Brownian motion.

2.3 Ito's formula (or lemma)

In this section we will derive Ito's formula; a more rigorous treatment can be found in Karatzas and Shreve (1991).

Let us consider the stochastic process X:

$$\mathrm{d}X = a\,\mathrm{d}t + b\,\mathrm{d}W = a\,\mathrm{d}t + b\sqrt{\mathrm{d}t}\,\mathrm{d}Z, \quad \mathrm{d}Z \sim \mathrm{N}(0,1), \mathrm{d}W \sim \mathrm{N}(0,\mathrm{d}t) \tag{2.3.1}$$

where a and b are constants. We want to find the process followed by a function of the stochastic variable X, that is $\phi(X,t)$. This can be done by applying a Taylor expansion, up to second order, in the two variables X and t as follows:

$$\phi^* = \phi + \frac{\partial\phi}{\partial t}\,\mathrm{d}t + \frac{\partial\phi}{\partial X}\,\mathrm{d}X + \frac{1}{2}\frac{\partial^2\phi}{\partial X^2}\,\mathrm{d}X^2 + \frac{1}{2}\frac{\partial^2\phi}{\partial t^2}\,\mathrm{d}t^2 + \frac{\partial\phi}{\partial X\,\partial t}\,\mathrm{d}X\,\mathrm{d}t \tag{2.3.2}$$

where ϕ^* is used to denote the value $\phi(X+\mathrm{d}X, t+\mathrm{d}t)$, and ϕ denotes the value $\phi(X,t)$. We will now consider the magnitude of the terms $\mathrm{d}X^2$, $\mathrm{d}X\,\mathrm{d}t$, and $\mathrm{d}t^2$ as $\mathrm{d}t \to 0$. First

$$\begin{aligned}\mathrm{d}X^2 &= \left(a\,\mathrm{d}t + b\sqrt{\mathrm{d}t}\,\mathrm{d}Z\right)\left(a\,\mathrm{d}t + b\sqrt{\mathrm{d}t}\,\mathrm{d}Z\right)\\ &= a^2\,\mathrm{d}t^2 + 2ab\,\mathrm{d}t^{3/2}\,\mathrm{d}Z + b^2\,\mathrm{d}t\,\mathrm{d}Z^2\end{aligned}$$

then

$$\mathrm{d}X\,\mathrm{d}t = a\,\mathrm{d}t^2 + b\,\mathrm{d}t^{3/2}\,\mathrm{d}Z$$

So as $\mathrm{d}t \to 0$, and ignoring all terms in $\mathrm{d}t$ of order greater than 1, we have:

$$\mathrm{d}X^2 \sim b^2\,\mathrm{d}t\,\mathrm{d}Z^2, \quad \mathrm{d}t^2 \sim 0, \quad \text{and} \quad \mathrm{d}X\,\mathrm{d}t \sim 0$$

Therefore Eq. (2.3.2) can be rewritten as:

$$\mathrm{d}\phi = \frac{\partial\phi}{\partial t}\,\mathrm{d}t + \frac{\partial\phi}{\partial X}\,\mathrm{d}X + \frac{1}{2}\frac{\partial^2\phi}{\partial X^2}E\left[\mathrm{d}X^2\right] \tag{2.3.3}$$

where $\mathrm{d}\phi = \phi^* - \phi$, and we have replaced $\mathrm{d}X^2$ by its expected value $E[\mathrm{d}X^2]$. Now

$$E\left[\mathrm{d}X^2\right] = E\left[b^2\,\mathrm{d}t\,\mathrm{d}Z^2\right] = b^2\,\mathrm{d}t\,E\left[\mathrm{d}Z^2\right] = b^2\,\mathrm{d}t$$

where we have used the fact that, since $\mathrm{d}Z \sim \mathrm{N}(0,1)$, the variance of $\mathrm{d}Z$, $E[\mathrm{d}Z^2]$, is by definition equal to 1. Using these values in Eq. (2.3.3) and substituting for $\mathrm{d}X$ from Eq. (2.3.1), we obtain:

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \frac{\partial \phi}{\partial X}(a\,\mathrm{d}t + b\,\mathrm{d}w) + \frac{b^2}{2}\frac{\partial^2 \phi}{\partial X^2}\,\mathrm{d}t \tag{2.3.4}$$

This gives Ito's formula

$$\mathrm{d}\phi = \left(\frac{\partial \phi}{\partial t} + a\frac{\partial \phi}{\partial X} + \frac{b^2}{2}\frac{\partial^2 \phi}{\partial X^2}\right)\mathrm{d}t + \frac{\partial \phi}{\partial X} b\,\mathrm{d}W \tag{2.3.5}$$

In particular if we consider the geometric Brownian process:

$$\mathrm{d}S = \mu S\,\mathrm{d}t + \sigma S\,\mathrm{d}W$$

where μ and σ are constants, then substituting $X = S$, $a = \mu S$, and $b = \sigma S$ into Eq. (2.3.5) yields:

$$\mathrm{d}\phi = \left(\frac{\partial \phi}{\partial t} + \mu S\frac{\partial \phi}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 \phi}{\partial S^2}\right)\mathrm{d}t + \frac{\partial \phi}{\partial S}\sigma S\,\mathrm{d}W \tag{2.3.6}$$

Equation (2.3.6) describes the change in value of a function $\phi(S,t)$ over the time interval $\mathrm{d}t$, when the stochastic variable S follows GBM. This result has very important applications in the pricing of financial derivatives. Here the function $\phi(S,t)$ is taken as the price of a financial derivative, $f(S,t)$, that depends on the value of an underlying asset S, which is assumed to follow GBM. In Chapter 4 we will use Eq. (2.3.6) to derive the (Black–Scholes) partial differential equation that is satisfied by the price of a financial derivative.

We can also use Eq. (2.3.3) to derive the process followed by $\phi = \log(S_t)$. We have:

$$\frac{\partial \phi}{\partial S_t} = \frac{\partial \log(S_t)}{\partial S} = \frac{1}{S}, \qquad \frac{\partial^2 \phi}{\partial S_t^2} = \frac{\partial}{\partial S_t}\left(\frac{\partial \log(S_t)}{\partial S_t}\right) = \frac{\partial}{\partial S_t}\left(\frac{1}{S_t}\right) = -\frac{1}{S_t^2}$$

$$\frac{\partial \phi}{\partial t} = \frac{\partial \log(S_t)}{\partial t} = 0$$

So

$$\mathrm{d}\big(\log(S_t)\big) = \nu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t \quad \text{where } \nu = \mu - \frac{\sigma^2}{2} \tag{2.3.7}$$

Integrating Eq. (2.3.7) yields

$$\int_{t=t_0}^{T} \mathrm{d}\big(\log(S_t)\big) = \int_{t=t_0}^{T} \nu\,\mathrm{d}t + \int_{t=t_0}^{T} \sigma\,\mathrm{d}W_t$$

so

$$\log(S_T) - \log(S_{t_0}) = \nu T + \sigma W_T \tag{2.3.8}$$

where we have used $t_0 = 0$ and $W_{t_0} = 0$.

We obtain

$$\log\left(\frac{S_T}{S_{t_0}}\right) \sim \mathrm{N}\left(\nu T, \sigma^2 T\right) \tag{2.3.9}$$

and so

$$\log\left(\frac{S_T}{S_{t_0}}\right) = \nu T + \sigma W_T \tag{2.3.10}$$

The solution to the geometric Brownian motion (GBM) in Eq. (2.2.3) is

$$S_T = S_{t_0} \exp(\nu T + \sigma W_T), \quad \nu = \mu - \frac{\sigma^2}{2} \tag{2.3.11}$$

The asset value at time $t + \mathrm{d}t$ can therefore be generated from its value at time t by using

$$S_{t+\mathrm{d}t} = S_t \exp\{\nu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t\}$$

We have shown that if the asset price follows geometric Brownian motion, then the logarithm of the asset price follows standard Brownian motion. Another way of stating this is that, over the time interval $\mathrm{d}t$, the change in the logarithm of the asset price is a Gaussian distribution with mean $(\mu - \sigma^2/2)\,\mathrm{d}t$, and variance $\sigma^2\,\mathrm{d}t$.

These results can easily be generalized to include time varying drift and volatility. Now instead of Eq. (2.2.3) we have

$$\mathrm{d}S_t = S_t\mu_t\,\mathrm{d}t + S_t\sigma_t\,\mathrm{d}W_t \tag{2.3.12}$$

which results in

$$\mathrm{d}\big(\log(S_t)\big) = \nu_t\,\mathrm{d}t + \sigma_t\,\mathrm{d}W_t \tag{2.3.13}$$

so

$$\int_{t=t_0}^{T} \mathrm{d}\big(\log(S_t)\big) = \int_{t=t_0}^{T} \nu_t\,\mathrm{d}t + \int_{t=t_0}^{T} \sigma_t\,\mathrm{d}W_t$$

which results in the following solution for S_T

$$S_T = S_{t_0} \exp\left(\int_{t=t_0}^{T} \nu_t\,\mathrm{d}t + \int_{t=t_0}^{T} \sigma_t\,\mathrm{d}W_t\right) \quad \text{where } \nu_t = \mu_t - \frac{\sigma_t^2}{2} \tag{2.3.14}$$

The results presented in Eqs. (2.3.11) and (2.3.14) are very important and will be referred to in later sections of the book.

2.4 Girsanov's theorem

This theorem states that for any stochastic process $k(t)$ such that $\int_0^t k(s)^2\,\mathrm{d}s < \infty$ then the Radon–Nikodym derivative $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} = \rho(t)$ is given by:

$$\rho(t) = \exp\left\{\int_0^t k(s)\,\mathrm{d}W_s^P - \frac{1}{2}\int_0^t k(s)^2\,\mathrm{d}s\right\} \tag{2.4.1}$$

where W_t^P is Brownian motion (possibly with drift) under probability measure $\mathbb{P}$, see Baxter and Rennie (1996). Under probability measure $\mathbb{Q}$ we have:

$$W_t^Q = W_t^P - \int_0^t k(s)\,\mathrm{d}s \tag{2.4.2}$$

where W_t^Q is also Brownian motion (possibly with drift).

We can also write

$$\mathrm{d}W^P = \mathrm{d}W^Q + k(t)\,\mathrm{d}t \tag{2.4.3}$$

Girsanov's theorem thus provides a mechanism for changing the drift of a Brownian motion.

2.5 Ito's lemma for multiasset geometric Brownian motion

We will now consider the n-dimensional stochastic process:

$$\mathrm{d}X_i = a_i\,\mathrm{d}t + b_i\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i = a_i\,\mathrm{d}t + b_i\,\mathrm{d}W_i, \quad i = 1, \ldots, n, \tag{2.5.1}$$

or in vector form:

$$\mathrm{d}X = A\,\mathrm{d}t + \sqrt{\mathrm{d}t}\,B\,\mathrm{d}Z = A\,\mathrm{d}t + B\,\mathrm{d}W \tag{2.5.2}$$

where A and B are n-element vectors respectively containing the constants, $a_i, i = 1, \ldots, n$, and $b_i, i = 1, \ldots, n$. The stochastic vector $\mathrm{d}X$ contains the n stochastic variables $X_i, i = 1, \ldots, n$.

We will assume that the n element random vector $\mathrm{d}Z$ is drawn from a multivariate normal distribution with zero mean and covariance matrix $\widehat{C}$. That is, we can write:

$$\mathrm{d}Z \sim \mathrm{N}(0, \widehat{C})$$

Since $\widehat{C}_{ii} = \mathrm{Var}[\mathrm{d}Z_i] = 1, i = 1, \ldots, n$, the diagonal elements of $\widehat{C}$ are all unity and the matrix $\widehat{C}$ is in fact a *correlation matrix* with off-diagonal elements given by:

$$\widehat{C}_{ij} = E[\mathrm{d}Z_i\,\mathrm{d}Z_j] = \rho_{i,j}, \quad i = 1, \ldots, n,\ j = 1, \ldots, n,\ i \neq j,$$

where ρ_{ij} is the correlation coefficient between the ith and jth elements of the vector $\mathrm{d}Z$.

Similarly the n-element random vector $\mathrm{d}W$ is drawn from a multivariate normal distribution with zero mean and covariance matrix C. We can thus write:

$$\mathrm{d}W \sim \mathrm{N}(0, C)$$

The diagonal elements of C are $C_{ii} = \mathrm{Var}[\mathrm{d}W_i] = \mathrm{d}t, i = 1, \ldots, n$, and off-diagonal elements are

$$C_{ij} = E[\mathrm{d}W_i\,\mathrm{d}W_j] = \rho_{i,j}\,\mathrm{d}t, \quad i = 1, \ldots, n,\ j = 1, \ldots, n,\ i \neq j$$

As in Section 2.3 we want to find the process followed by a function of the stochastic vector X, that is the process followed by $\phi(X, t)$. This can be done by applying an n-dimensional Taylor expansion, up to second order, in the variables X and t as follows:

$$\phi^* = \phi + \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \sum_{i=1}^{n} \frac{\partial \phi}{\partial X_i}\,\mathrm{d}X_i + \frac{1}{2} E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 \phi}{\partial X_i\,\partial X_j}\,\mathrm{d}X_i\,\mathrm{d}X_j\right] + \frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}\,\mathrm{d}t^2 + \sum_{i=1}^{n} \frac{\partial \phi}{\partial X_i\,\partial t}\,\mathrm{d}X_i\,\mathrm{d}t \tag{2.5.3}$$

where ϕ^* is used to denote the value $\phi(X + \mathrm{d}X, t + \mathrm{d}t)$, and ϕ denotes the value $\phi(X, t)$. We will now consider the magnitude of the terms $\mathrm{d}X_i\,\mathrm{d}X_j$, $\mathrm{d}X_i\,\mathrm{d}t$, and $\mathrm{d}t^2$ as $\mathrm{d}t \to 0$. Expanding the terms $\mathrm{d}X_i\,\mathrm{d}X_j$ and $\mathrm{d}X_i\,\mathrm{d}t$ we have:

$$\begin{aligned}
&\mathrm{d}X_i\,\mathrm{d}X_j = \left(a_i\,\mathrm{d}t + b_i\sqrt{\mathrm{d}t}\,\mathrm{d}Z_i\right)\left(a_j\,\mathrm{d}t + b_j\sqrt{\mathrm{d}t}\,\mathrm{d}Z_j\right)\\
&\therefore \mathrm{d}X_i\,\mathrm{d}X_j = a_i a_j\,\mathrm{d}t^2 + a_i b_j\,\mathrm{d}t^{3/2}\,\mathrm{d}Z_j + a_j b_i\,\mathrm{d}t^{3/2}\,\mathrm{d}Z_i\\
&\qquad\qquad + b_i b_j\,\mathrm{d}t\,\mathrm{d}Z_i\,\mathrm{d}Z_j\\
&\mathrm{d}X_i\,\mathrm{d}t = a_i\,\mathrm{d}t^2 + b_i\,\mathrm{d}t^{3/2}\,\mathrm{d}Z_i
\end{aligned} \tag{2.5.4}$$

So as $\mathrm{d}t \to 0$, and ignoring all terms in $\mathrm{d}t$ of order greater than 1, we have:

$$\mathrm{d}X_i\,\mathrm{d}t \sim 0$$

and

$$\mathrm{d}X_i\,\mathrm{d}X_j \sim b_i b_j\,\mathrm{d}t\,\mathrm{d}Z_i\,\mathrm{d}Z_j$$

Therefore Eq. (2.5.3) can be rewritten as

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \sum_{i=1}^{n} \frac{\partial \phi}{\partial X_i}\,\mathrm{d}X_i + \frac{1}{2} E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 \phi}{\partial X_i\,\partial X_j}\,\mathrm{d}X_i\,\mathrm{d}X_j\right] \tag{2.5.5}$$

where $\mathrm{d}\phi = \phi^* - \phi$.

Now

$$E[\mathrm{d}X_i\,\mathrm{d}X_j] = E[b_i b_j\,\mathrm{d}t\,\mathrm{d}Z_i\,\mathrm{d}Z_j] = b_i b_j\,\mathrm{d}t\,E[\mathrm{d}Z_i\,\mathrm{d}Z_j] = b_i b_j \rho_{ij}\,\mathrm{d}t$$

where ρ_{ij} is the correlation coefficient between the ith and jth assets.

Using these values in Eq. (2.5.5), and substituting for $\mathrm{d}X_i$ from Eq. (2.5.1), we obtain:

$$\mathrm{d}\phi = \sum_{i=1}^{n} \frac{\partial \phi}{\partial X_i}(a_i\,\mathrm{d}t + b_i\,\mathrm{d}W_i) + \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_i b_j \rho_{ij}\,\mathrm{d}t\,\frac{\partial^2 \phi}{\partial X_i\,\partial X_j} \tag{2.5.6}$$

This gives Ito's n-dimensional formula:

$$d\phi = \left\{ \frac{\partial \phi}{\partial t} + \sum_{i=1}^{n} a_i \frac{\partial \phi}{\partial X_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_i b_j \rho_{ij} \frac{\partial^2 \phi}{\partial X_i \, \partial X_j} \right\} dt + \sum_{i=1}^{n} \frac{\partial \phi}{\partial X_i} b_i \, dW_i \tag{2.5.7}$$

In particular if we consider the geometric Brownian motion:

$$dS_i = \mu_i S_i \, dt + \sigma_i S_i \, dW_i, \quad i = 1, \ldots, n,$$

where μ_i is the constant drift of the ith asset and σ_i is the constant volatility of the ith asset, then substituting $X_i = S_i$, $a_i = \mu_i S_i$, and $b_i = \sigma_i S_i$ into Eq. (2.5.7) yields:

$$d\phi = \left\{ \frac{\partial \phi}{\partial t} + \sum_{i=1}^{n} \mu_i S_i \frac{\partial \phi}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_i \sigma_j S_i S_j \rho_{ij} \frac{\partial^2 \phi}{\partial S_i \, \partial S_j} \right\} dt + \sum_{i=1}^{n} \frac{\partial \phi}{\partial S_i} \sigma_i S_i \, dW_i \tag{2.5.8}$$

2.6 Ito product and quotient rules in two dimensions

We will now derive expressions for the product and quotient of two stochastic processes. In this case $\phi \to \phi(X_1, X_2)$, with

$$dX_1 = a_1 \, dt + b_1 \, dW_1 \quad \text{and} \quad dX_2 = a_2 \, dt + b_2 \, dW_2$$

The following two-dimensional version of Ito's lemma will be used:

$$d\phi = \frac{\partial \phi}{\partial X_1} dX_1 + \frac{\partial \phi}{\partial X_2} dX_2 + \frac{1}{2} E\left[\sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial^2 \phi}{\partial X_i \, \partial X_j} dX_i \, dX_j \right] \tag{2.6.1}$$

where we have used the fact that $\frac{\partial \phi}{\partial t} = 0$.

2.6.1 Ito product rule

Here $\phi = \phi(X_1 X_2)$, and the partial derivatives are as follows:

$$\frac{\partial \phi}{\partial X_1} = X_2, \qquad \frac{\partial \phi}{\partial X_2} = X_1 \frac{\partial^2 \phi}{\partial X_1^2} = \frac{\partial^2 \phi}{\partial X_2^2} = 0,$$

$$\frac{\partial^2 \phi}{\partial X_1 \, \partial X_2} = \frac{\partial^2 \phi}{\partial X_2 \, \partial X_1} = 1$$

Therefore using Eq. (2.6.1)

$$d\phi = X_2 \, dX_1 + X_1 \, dX_2 + \frac{2E[dX_1 \, dX_2]}{2}$$

and the product rule is

$$\mathrm{d}(X_1X_2) = X_2\,\mathrm{d}X_1 + X_1\,\mathrm{d}X_2 + E[\mathrm{d}X_1\,\mathrm{d}X_2] \tag{2.6.2}$$

Brownian motion with one source of randomness

For the special case where X_1 is Brownian motion and X_2 has no random term we have:

$$\mathrm{d}X_1 = X_1\mu_1\,\mathrm{d}t + X_1\sigma_1\,\mathrm{d}W_1 \quad \text{and} \quad \mathrm{d}X_2 = X_2\mu_2\,\mathrm{d}t$$

Now

$$\begin{aligned} E[\mathrm{d}X_1\,\mathrm{d}X_2] &= E\big[(X_1\mu_1\,\mathrm{d}t + X_1\sigma_1\,\mathrm{d}W_1)X_2\mu_2\,\mathrm{d}t\big] \\ &= X_1X_2\mu_1\mu_2\,\mathrm{d}t^2 + X_1X_2\sigma_1\mu_2\,\mathrm{d}t\,X_2\mu_2\,\mathrm{d}t\,E[\mathrm{d}W_1] \\ &= 0 \end{aligned}$$

where we have ignored terms in $\mathrm{d}t$ with order higher than 1, and used $E[\mathrm{d}W_1] = 0$.

Therefore Eq. (2.6.2) becomes:

$$\mathrm{d}(X_1X_2) = X_2\,\mathrm{d}X_1 + X_1\,\mathrm{d}X_2$$
$$\mathrm{d}(X_1X_2) = X_2(X_1\mu_1\,\mathrm{d}t + X_1\sigma_1\,\mathrm{d}W_1) + X_1X_2\mu_2\,\mathrm{d}t$$

So we finally obtain:

$$\mathrm{d}(X_1X_2) = (X_1X_2)\{\mu_1 + \mu_2\}\,\mathrm{d}t + (X_1X_2)\sigma_1\,\mathrm{d}W_1 \tag{2.6.3}$$

2.6.2 Ito quotient rule

Here $\phi = \phi(X_1/X_2)$ and the partial derivatives are as follows:

$$\frac{\partial\phi}{\partial X_1} = \frac{1}{X_2}, \qquad \frac{\partial\phi}{\partial X_2} = -\frac{X_1}{X_2^2}, \qquad \frac{\partial^2\phi}{\partial X_1^2} = 0,$$
$$\frac{\partial^2\phi}{\partial X_2^2} = 2\frac{X_1}{X_2^3}, \qquad \frac{\partial^2\phi}{\partial X_1\,\partial X_2} = \frac{\partial^2\phi}{\partial X_2\,\partial X_1} = -\frac{1}{X_2^2}$$

Therefore using Eq. (2.6.1)

$$\mathrm{d}\phi = \frac{\mathrm{d}X_1}{X_2} - X_1\frac{\mathrm{d}X_2}{X_2^2} + \frac{1}{2}E\left[\left\{\frac{2X_1}{X_2^3}\,\mathrm{d}X_2^2 - 2\frac{\mathrm{d}X_1\,\mathrm{d}X_2}{X_2^2}\right\}\right]$$

We obtain the following expression for the quotient rule:

$$\begin{aligned} \mathrm{d}\left(\frac{X_1}{X_2}\right) = \left(\frac{X_1}{X_2}\right)\Bigg\{&\frac{\mathrm{d}X_1}{X_1} - \frac{\mathrm{d}X_2}{X_2} + E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] \\ &- E\left[\left(\frac{\mathrm{d}X_1}{X_1}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right]\Bigg\} \end{aligned} \tag{2.6.4}$$

Brownian motion

Here we have:

$$\mathrm{d}X_1 = X_1\mu_1\,\mathrm{d}t + X_1\sigma_1\,\mathrm{d}W_1 \quad \text{and} \quad \mathrm{d}X_2 = X_2\mu_2\,\mathrm{d}t + X_2\sigma_2\,\mathrm{d}W_2$$

or equivalently

$$\frac{\mathrm{d}X_1}{X_1} = \mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}W_1 \quad \text{and} \quad \frac{\mathrm{d}X_2}{X_2} = \mu_2\,\mathrm{d}t + \sigma_2\,\mathrm{d}W_2$$

Therefore

$$\begin{aligned} E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] &= E\big[(\mu_2\,\mathrm{d}t + \sigma_2\,\mathrm{d}W_2)(\mu_2\,\mathrm{d}t + \sigma_2\,\mathrm{d}W_2)\big] \\ &= E\big[\mu_2^2\,\mathrm{d}t^2\big] + E\big[\sigma_2^2(\mathrm{d}W_2)^2\big] + 2E[\sigma_2\,\mathrm{d}t\,\mathrm{d}W_2] \\ &= \mu_2^2\,\mathrm{d}t^2 + \sigma_2^2\,\mathrm{d}t + 2\sigma_2\,\mathrm{d}t\,E[\mathrm{d}W_2] \end{aligned}$$

which results in

$$E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] = \sigma_2^2\,\mathrm{d}t \tag{2.6.5}$$

where we have ignored all terms in $\mathrm{d}t$ with order higher than 1, and used the fact that $E[\mathrm{d}W_2] = 0$.

In a similar manner

$$\begin{aligned} E\left[\left(\frac{\mathrm{d}X_1}{X_1}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] &= E\big[(\mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}W_1)(\mu_2\,\mathrm{d}t + \sigma_2\,\mathrm{d}W_2)\big] \\ &= E\big[\mu_1\mu_2\,\mathrm{d}t^2\big] + E[\sigma_1\mu_2\,\mathrm{d}t\,\mathrm{d}W_1] \\ &\quad + E[\sigma_2\mu_1\,\mathrm{d}t\,\mathrm{d}W_2] + E[\sigma_1\sigma_2\,\mathrm{d}W_1\,\mathrm{d}W_2] \\ &= \mu_1\mu_2\,\mathrm{d}t^2 + \sigma_1\mu_2\,\mathrm{d}t\,E[\mathrm{d}W_1] + \sigma_2\mu_1\,\mathrm{d}t\,E[\mathrm{d}W_2] \\ &\quad + \sigma_1\sigma_2 E[\mathrm{d}W_1\,\mathrm{d}W_2] \end{aligned}$$

which gives

$$E\left[\left(\frac{\mathrm{d}X_1}{X_1}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] = \sigma_1\sigma_2\,\mathrm{d}t\,\rho_{12} \tag{2.6.6}$$

where we have proceeded as before but also used the fact that $E[\mathrm{d}W_1\,\mathrm{d}W_2] = \rho_{12}\,\mathrm{d}t$.

Substituting these into Eq. (2.6.4) we have:

$$\begin{aligned} \mathrm{d}\left(\frac{X_1}{X_2}\right) &= \left(\frac{X_1}{X_2}\right)\left\{\frac{\mathrm{d}X_1}{X_1} - \frac{\mathrm{d}X_2}{X_2} + \sigma_2^2\,\mathrm{d}t - \sigma_1\sigma_2\rho_{12}\,\mathrm{d}t\right\} \\ &= \left(\frac{X_1}{X_2}\right)\{\mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}t - \mu_2\,\mathrm{d}t - \sigma_2\,\mathrm{d}t + \sigma_2^2\,\mathrm{d}t - \sigma_1\sigma_2\rho_{12}\,\mathrm{d}t\} \\ &= \left(\frac{X_1}{X_2}\right)\{\mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}W_1 - \mu_2\,\mathrm{d}t - \sigma_2\,\mathrm{d}W_2 + \sigma_2^2\,\mathrm{d}t - \sigma_1\sigma_2\rho_{12}\,\mathrm{d}t\} \end{aligned}$$

This yields:

$$\mathrm{d}\left(\frac{X_1}{X_2}\right) = \left(\frac{X_1}{X_2}\right)\{\mu_1 - \mu_2 + \sigma_2^2 - \sigma_1\sigma_2\rho_{12}\}\,\mathrm{d}t + \left(\frac{X_1}{X_2}\right)\{\sigma_1\,\mathrm{d}W_1 - \sigma_2\,\mathrm{d}W_2\} \tag{2.6.7}$$

Brownian motion with one source of randomness

We have

$$\mathrm{d}X_1 = X_1\mu_1\,\mathrm{d}t + X_1\sigma_1\,\mathrm{d}W_1 \quad \text{and} \quad \mathrm{d}X_2 = X_2\mu_2\,\mathrm{d}t$$

As before

$$E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] = E\left[\mu_2^2\,\mathrm{d}t^2\right] = \mu_2^2\,\mathrm{d}t^2 \to 0$$

$$\begin{aligned} E\left[\left(\frac{\mathrm{d}X_1}{X_1}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] &= E\left[(\mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}W_1)\mu_2\,\mathrm{d}t\right] \\ &= E\left[\mu_1\mu_2\,\mathrm{d}t^2 + \sigma_1\mu_2\,\mathrm{d}t\,\mathrm{d}W_1\right] \\ &= \mu_1\mu_2\,\mathrm{d}t^2 + \sigma_1\mu_2\,\mathrm{d}t\,E[\mathrm{d}W_1] \to 0 \end{aligned}$$

Therefore

$$\begin{aligned} \mathrm{d}\left(\frac{X_1}{X_2}\right) &= \left(\frac{X_1}{X_2}\right)\left\{\frac{\mathrm{d}X_1}{X_1} - \frac{\mathrm{d}X_2}{X_2}\right\} \\ &= \left(\frac{X_1}{X_2}\right)\{\mu_1\,\mathrm{d}t + \sigma_1\,\mathrm{d}W_1 - \mu_2\,\mathrm{d}t\} \end{aligned}$$

So the final expression is

$$\mathrm{d}\left(\frac{X_1}{X_2}\right) = \left(\frac{X_1}{X_2}\right)\{\mu_1 - \mu_2\}\,\mathrm{d}t + \left(\frac{X_1}{X_2}\right)\sigma_1\,\mathrm{d}W_1 \tag{2.6.8}$$

2.7 Ito product in n dimensions

Using Eq. (2.5.7) we will now derive an expression for the product of n stochastic processes. In this case $\phi \to \prod_{i=1}^{n} X_i$, and the partial derivatives are as follows:

$$\frac{\partial\phi}{\partial X_i} = \phi\frac{\mathrm{d}X_i}{X_i} \quad \text{for } i = 1, \ldots, n$$

$$\frac{\partial^2\phi}{\partial X_i^2} = 0 \quad \text{for } i = 1, \ldots, n$$

$$\frac{\partial^2\phi}{\partial X_i\,\partial X_j} = \frac{\partial^2\phi}{\partial X_j\,\partial X_i}$$
$$= \left(\frac{\mathrm{d}X_i}{X_i}\right)\left(\frac{\mathrm{d}X_j}{X_j}\right)\phi \quad \text{for } i \neq j, i = 1, \ldots, n, j = 1, \ldots, n$$
$$\frac{\partial\phi}{\partial t} = 0$$

So substituting into Eq. (2.5.7) we have

$$\mathrm{d}\phi = \phi\sum_{i=1}^{n}\left(\frac{\mathrm{d}X_i}{X_i}\right) + \phi E\left[\sum_{i=1}^{n}\sum_{j=1(i\neq j)}^{n}\left(\frac{\mathrm{d}X_i}{X_i}\right)\left(\frac{\mathrm{d}X_j}{X_j}\right)\right] \tag{2.7.1}$$

which in full is

$$\mathrm{d}\left(\prod_{i=1}^{n}X_i\right) = \left(\prod_{i=1}^{n}X_i\right)\sum_{i=1}^{n}\left(\frac{\mathrm{d}X_i}{X_i}\right)$$
$$+ \left(\prod_{i=1}^{n}X_i\right)E\left[\sum_{i=1}^{n}\sum_{j=1(i\neq j)}^{n}\left(\frac{\mathrm{d}X_i}{X_i}\right)\left(\frac{\mathrm{d}X_j}{X_j}\right)\right] \tag{2.7.2}$$

2.8 The Brownian bridge

Let a Brownian process have values W_{t_0} at time t_0 and W_{t_1} at time t_1. We want to find the conditional distribution of W_t, where $t_0 < t < t_1$. This distribution will be denoted by $P(W_t|\{W_{t_0}, W_{t_1}\})$, to indicate that W_t is conditional on the end values W_{t_0} and W_{t_1}. We now write W_{t_0} and W_{t_1} as

$$W_t = W_{t_0} + \sqrt{t - t_0}X_t, \quad X_t \sim \mathrm{N}(0, 1), \tag{2.8.1}$$
$$W_{t_1} = W_t + \sqrt{t_1 - t}Y_t, \quad Y_t \sim \mathrm{N}(0, 1), \tag{2.8.2}$$

where X_t and Y_t are independent normal variates.

Combining Eqs. (2.8.1) and (2.8.2) we have

$$W_{t_1} = W_{t_0} + \sqrt{t - t_0}X_t + \sqrt{t_1 - t}Y_t$$

which can be re-expressed as

$$W_{t_1} - W_{t_0} = \sqrt{t - t_0}X_t + \sqrt{t_1 - t}Y_t$$

Using the Brownian motion property (iii) in Section 2.1

$$W_{t_1} - W_{t_0} = \sqrt{t_1 - t_0}Z_t, \quad Z_t \sim \mathrm{N}(0, 1)$$

So

$$\sqrt{t_1 - t_0}Z_t = \sqrt{t - t_0}X_t + \sqrt{t_1 - t}Y_t$$

and

$$Y(X_t, Z_t) = \frac{\sqrt{t_1 - t_0}Z_t - \sqrt{t - t_0}X_t}{\sqrt{t_1 - t}} \tag{2.8.3}$$

Now $P(W_t|\{W_{t_0}, W_{t_1}\}) = P(X_t|Z_t)$, the probability distribution of X_t conditional on Z_t. From Bayes law

$$P(X_t|Z_t) = \frac{P(X_t)P(Y(X_t, Z_t))}{P(Z_t)} = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{X_t^2 + Y_t^2 - Z_t^2}{2}\right\} \tag{2.8.4}$$

Since X_t, Y_t and Z_t are Gaussians we can write

$$P(X_t|Z_t) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{X_t^2 + Y_t^2 - Z_t^2}{2}\right\} \tag{2.8.5}$$

First let us compute Y_t^2.

$$Y_t^2 = \left(\frac{\sqrt{t_1 - t_0}Z_t - \sqrt{t - t_0}X_t}{\sqrt{t_1 - t}}\right)^2$$

so

$$Y_t^2 = \frac{(t_1 - t_0)Z_t^2 + (t - t_0)X_t^2 - 2\sqrt{t_1 - t_0}\sqrt{t - t_0}X_t Z_t}{t_1 - t} \tag{2.8.6}$$

Next we compute $X_t^2 + Y_t^2 - Z_t^2$ as follows

$$\begin{aligned} &X_t^2 + Y_t^2 - Z_t^2 \\ &\quad = \frac{(t_1 - t_0)X_t^2 + (t - t_0)Z_t^2 - 2\sqrt{t_1 - t_0}\sqrt{t - t_0}X_t Z_t}{t_1 - t} \end{aligned} \tag{2.8.7}$$

Dividing top and bottom of Eq. (2.8.7) by $t_1 - t_0$ we obtain:

$$\begin{aligned} &X_t^2 + Y_t^2 - Z_t^2 \\ &\quad = \frac{X_t^2 + ((t - t_0)/(t_1 - t_0))Z_t^2 - 2\sqrt{t_1 - t_0}\sqrt{t - t_0}/(t_1 - t_0)X_t Z_t}{(t_1 - t)/(t_1 - t_0)} \\ &\quad = \frac{X_t^2 + ((t - t_0)/(t_1 - t_0))Z_t^2 - 2\sqrt{(t - t_0)/(t_1 - t_0)}X_t Z_t}{(t_1 - t)/(t_1 - t_0)} \end{aligned}$$

which gives

$$X_t^2 + Y_t^2 - Z_t^2 = \frac{(X_t - \sqrt{(t - t_0)/(t_1 - t_0)}Z_t)^2}{(t_1 - t)/(t_1 - t_0)} \tag{2.8.8}$$

where we have used

$$\left(X_t - \sqrt{\frac{t - t_0}{t_1 - t_0}}Z_t\right)^2 = X_t^2 + \frac{t - t_0}{t_1 - t_0}Z_t^2 - 2\sqrt{\frac{t - t_0}{t_1 - t_0}}X_t Z_t$$

Substituting Eq. (2.8.8) into Eq. (2.8.5) yields

$$P(X_t|Z_t) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(X_t - \sqrt{(t - t_0)/(t_1 - t_0)}Z_t)^2}{2(t_1 - t)/(t_1 - t_0)}\right\}$$

Therefore $P(X_t|Z_t)$ is a Gaussian distribution with:

$$E[X_t] = \sqrt{\frac{t-t_0}{t_1-t_0}}Z_t \quad \text{and} \quad \text{Var}[X_t] = \frac{t_1-t}{t_1-t_0}$$

Substituting for Z_t we have

$$E[X_t] = \sqrt{\frac{t-t_0}{t_1-t_0}}Z_t = \sqrt{\frac{t-t_0}{t_1-t_0}}\frac{W_{t_1}-W_{t_0}}{\sqrt{t_1-t_0}}$$

which gives:

$$E[X_t] = \frac{\sqrt{t-t_0}}{t_1-t_0}(W_{t_1}-W_{t_0}) \tag{2.8.9}$$

The variate $\widehat{X_t} = E[X_t] + \sqrt{\text{Var}[X_t]}Z_t$ has the same distribution as $P(X_t|Z_t)$.

So we can substitute $\widehat{X_t}$ for X_t in Eq. (2.8.1) to obtain:

$$W_t = W_{t_0} + \sqrt{t-t_0}\left\{E[X_t] + \sqrt{\text{Var}[X_t]}Z_t\right\}$$

which gives:

$$W_t = W_{t_0} + \sqrt{t-t_0}\left\{\frac{\sqrt{t-t_0}}{t_1-t_0}(W_{t_1}-W_{t_0}) + \sqrt{\frac{t_1-t}{t_1-t_0}}Z_t\right\}$$

and simplifying we obtain:

$$W_t = W_{t_0}\frac{t_1-t_0}{t_1-t_0} + \frac{t-t_0}{t_1-t_0}(W_{t_1}-W_{t_0}) + \sqrt{\frac{(t_1-t)(t-t_0)}{t_1-t_0}}Z_t \tag{2.8.10}$$

Variates, W_t, from the distribution of $P(W_t|\{W_{t_0}, W_{t_1}\})$ can therefore be generated by using

$$W_t = W_{t_0}\frac{t_1-t}{t_1-t_0} + W_{t_1}\frac{t-t_0}{t_1-t_0} + \sqrt{\frac{(t_1-t)(t-t_0)}{t_1-t_0}}Z_t \tag{2.8.11}$$

An alternative derivation of the Brownian bridge is given in Appendix G.

2.9 Time-transformed Brownian motion

Let us consider the Brownian motion:

$$\mathrm{d}W_t = \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_t \tag{2.9.1}$$

and also the scaled and time-transformed Brownian motion

$$Y_{W,t} = a_t W_{f_t} \tag{2.9.2}$$

where the scale factor, a_t, is a real function and the time transformation, f_t, is a continuous increasing function satisfying $f_t \geqslant 0$; see Cox and Miller (1965).

Using Ito's lemma,

$$\mathrm{d}Y_{W,t} = \frac{\partial Y_t}{\partial t}\,\mathrm{d}t + \frac{\partial Y_t}{\partial W_t}\,\mathrm{d}W_t \tag{2.9.3}$$

where we have used the fact that $\frac{\partial^2 Y_t}{\partial W_t^2} = 0$.

From Eq. (2.9.3)

$$\mathrm{d}Y_{W,t} = \left(\frac{\partial a_t}{\partial t}\right) W_{f_t}\,\mathrm{d}t + a_t\,\mathrm{d}W_{f_t} \tag{2.9.4}$$

Now

$$\mathrm{d}W_{f_t} = \sqrt{\mathrm{d}f_t}\,\mathrm{d}Z_t = \sqrt{\frac{\partial f_t}{\partial t}\,\mathrm{d}t}\,\mathrm{d}Z_t \tag{2.9.5}$$

so we can write:

$$\mathrm{d}Y_{W,t} = a_t' W_{f_t}\,\mathrm{d}t + a_t\sqrt{f_t'\,\mathrm{d}t}\,\mathrm{d}Z_t \tag{2.9.6}$$

where

$$a_t' = \frac{\partial a_t}{\partial t} \quad \text{and} \quad f_t' = \frac{\partial f_t}{\partial t}$$

2.9.1 Scaled Brownian motion

We will prove that $\widehat{W_t}$ defined by

$$\widehat{W_t} = \frac{1}{c} W_{c^2 t}, \quad c > 0,$$

is Brownian motion.

Let us consider the process

$$Y_t = W_{c^2 t}$$

From Eq. (2.9.2) we have $a_t = 1$, $f_t = c^2 t$, $a_t' = 0$ and $f_t' = c^2$. Substituting these values into Eq. (2.9.6), yields

$$\mathrm{d}Y_t = \sqrt{c^2 \mathrm{d}t}\,\mathrm{d}Z_t$$

which gives

$$\mathrm{d}Y_t = c\sqrt{\mathrm{d}t}\,\mathrm{d}Z_t = c\,\mathrm{d}W_t$$

Therefore $\widehat{W_t} = \mathrm{d}Y_t/c$ is Brownian motion.

2.9.2 The Ornstein–Uhlenbeck process

We will now show that the Ornstein–Uhlenbeck process (see Section 2.10) can be represented as follows:

$$Y_{W,t} = \exp(-\alpha t) W_{\psi_t} \quad \text{where } \psi_t = \frac{\sigma^2 \exp(2\alpha t)}{2\alpha}, \alpha > 0 \tag{2.9.7}$$

PROOF. From Eqs. (2.9.2) and (2.9.7) we have:

$$f_t = \frac{\sigma^2 \exp(\sigma^2 \exp(2\alpha t))}{2\alpha} \quad \text{and} \quad a_t = \exp(-2\alpha t) \tag{2.9.8}$$

Therefore

$$\frac{a_t'}{a_t} = -\frac{-\alpha \exp(-\alpha t)}{\exp(-\alpha t)} = -\alpha \tag{2.9.9}$$

and

$$f_t' = \frac{\sigma^2}{2\alpha} 2\alpha \exp(2\alpha t) = \sigma^2 \exp(2\alpha t) \tag{2.9.10}$$

So

$$\sqrt{f_t' \, \mathrm{d}t} = \sqrt{\sigma^2 \exp(2\alpha t)} = \sigma \exp(\alpha t)\sqrt{\mathrm{d}t} \tag{2.9.11}$$

Thus

$$\mathrm{d}Y_{W,t} = -\alpha Y_{W,t} \, \mathrm{d}t + \exp(-\alpha t)\sigma \exp(\alpha t)\sqrt{\mathrm{d}t} \, \mathrm{d}Z \tag{2.9.12}$$

which means that

$$\mathrm{d}Y_{W,t} = -\alpha Y_{W,t} \, \mathrm{d}t + \sigma \, \mathrm{d}W_t \tag{2.9.13}$$

From Eq. (2.9.13) it can be seen that conditional mean and variance are

$$E[\mathrm{d}Y_{W,t} | F_t] = \alpha Y_{W,t} \, \mathrm{d}t \tag{2.9.14}$$

$$\mathrm{Var}[\mathrm{d}Y_{W,t} | F_t] = \sigma^2 \, \mathrm{d}t \tag{2.9.15}$$

□

Unconditional mean

The unconditional mean is

$$E[Y_{W,t}] = E\left[\exp(-\alpha t) W\left(\frac{\sigma^2 \exp(2\alpha t)}{2\alpha}\right)\right] \quad \text{where } \alpha > 0 \text{ and } t \to \infty \tag{2.9.16}$$

So

$$E[Y_{W,t}] = 0 \tag{2.9.17}$$

Unconditional variance and covariance

Let

$$Y_{W,t} = \exp(-\alpha t) W_{\psi_t} \quad \text{where } \psi_t = \left(\frac{\sigma^2 \exp(2\alpha t)}{2\alpha}\right) \tag{2.9.18}$$

and

$$Y_{W,s} = \exp(-\alpha s) W_{\psi_s} \quad \text{where } \psi_s = \left(\frac{\sigma^2 \exp(2\alpha s)}{2\alpha} \right) \tag{2.9.19}$$

The covariance is:

$$\begin{aligned} \text{Cov}[Y_{W,s}, Y_{W,t}] &= E[Y_{W,t} Y_{W,s}] - E[Y_{W,t}] E[Y_{W,s}] \\ &= E[Y_{W,t} Y_{W,s}] \end{aligned} \tag{2.9.20}$$

since $E[Y_{W,s}] = E[Y_{W,t}] = 0$.

Shortening the notation of $Y_{W,t}$ to Y_t we obtain

$$\begin{aligned} \text{Cov}[Y_s, Y_t] &= E\big[\exp(-\alpha t) W_{\psi_t} \exp(-\alpha t) W_{\psi_s}\big] \\ &= \exp\big(-\alpha(t+s)\big) E\big[\{W_{\psi_t} W_{\psi_s}\}\big] \end{aligned}$$

From Eq. (2.1.2)

$$E[W_s, W_t] = s \wedge t \tag{2.9.21}$$

Therefore, if $s \leqslant t$

$$E[W_{\psi_t} W_{\psi_s}] = W_{\psi_s} \tag{2.9.22}$$

and

$$\text{Cov}[Y_s, Y_t] = \frac{\exp(-\alpha(t+s))\sigma^2 \exp(2\alpha s)}{2\alpha} = \frac{\sigma^2}{2\alpha} \exp\big(-\alpha(t-s)\big) \tag{2.9.23}$$

The unconditional variance (obtained by setting $s = t$) is

$$\text{Var}[Y_t] = \frac{\sigma^2}{2\alpha} \tag{2.9.24}$$

2.10 Ornstein–Uhlenbeck process

The Ornstein–Uhlenbeck process is often used to model interest rates because of its mean reverting property. It is defined by the equation

$$\mathrm{d}X_t = -\alpha X_t \, \mathrm{d}t + \sigma \, \mathrm{d}W_t \tag{2.10.1}$$

Using the integrating factor $\exp(\alpha t)$ we have:

$$\exp(\alpha t) \, \mathrm{d}X_t = -\alpha X_t \exp(\alpha t) \, \mathrm{d}t + \sigma \exp(\alpha t) \, \mathrm{d}W_t$$

so

$$\exp(\alpha t) \, \mathrm{d}X_t + \alpha X_t \exp(\alpha t) \, \mathrm{d}t = \sigma \exp(\alpha t) \, \mathrm{d}W_t \tag{2.10.2}$$

Using the Ito product rule we have:

$$\mathrm{d}\big(X_t \exp(\alpha t)\big) = \exp(\alpha t) \, \mathrm{d}X_t + \alpha X_t \exp(\alpha t) \, \mathrm{d}t \tag{2.10.3}$$

So from Eqs. (2.10.2) and (2.10.3) we obtain

$$\mathrm{d}\big(X_t \exp(\alpha t)\big) = \sigma \exp(\alpha t)\,\mathrm{d}W_t \tag{2.10.4}$$

Integrating Eq. (2.10.4) gives

$$\int_{s=0}^{s=t} \mathrm{d}\big(X_s \exp(\alpha s)\big) = \sigma \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s$$

which yields

$$X_t \exp(\alpha t) - X_{t_0} = \sigma \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s$$

and thus the solution of Eq. (2.10.1) is

$$X_t = X_{t_0} \exp(-\alpha t) + \sigma \exp(-\alpha t) \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s \tag{2.10.5}$$

We will now derive expressions for both the unconditional mean and the unconditional variance of X_t.

The mean

Taking expectations of both sides of Eq. (2.10.5) yields

$$E[X_t] = E\big[X_{t_0} \exp(-\alpha t)\big] + E\left[\sigma \exp(-\alpha t) \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right] \tag{2.10.6}$$

since

$$E\left[\sigma \exp(-\alpha t) \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right]$$
$$= \sigma \exp(-\alpha t) E\left[\int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right] = 0$$

the unconditional mean is

$$E[X_t] = X_{t_0} \exp(-\alpha t) \tag{2.10.7}$$

The variance

To derive the expression for unconditional variance requires a bit more effort. We have

$$\begin{aligned} \mathrm{Var}[X_t] &= E\big[\{X_t - E[X_t]\}^2\big] \\ &= E\big[\{X_t - X_{t_0} \exp(-\alpha t)\}^2\big] \end{aligned} \tag{2.10.8}$$

However, from Eq. (2.10.5)

$$X_t - X_{t_0} \exp(-\alpha t) = \sigma \exp(-\alpha t) \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s$$

So substituting the above expression into Eq. (2.10.8)

$$\mathrm{Var}[X_t] = E\left[\left\{\exp(-\alpha t)\sigma \int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right\}^2\right] \tag{2.10.9}$$

$$\mathrm{Var}[X_t] = \sigma^2 \exp(-2\alpha t) E\left[\left\{\int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right\}^2\right] \tag{2.10.10}$$

Using Ito's isometry (see Section 2.12.2)

$$E\left[\left\{\int_{s=0}^{s=t} \exp(\alpha s)\,\mathrm{d}W_s\right\}^2\right] = E\left[\int_{s=0}^{s=t} \left\{\exp(\alpha s)\right\}^2 \mathrm{d}s\right]$$

Then using Fubini's theorem (see Section 2.12.1)

$$\begin{aligned} E\left[\int_{s=0}^{s=t} \left\{\exp(\alpha s)\right\}^2 \mathrm{d}s\right] &= \int_{s=0}^{s=t} E\left[\left\{\exp(\alpha s)\right\}^2\right] \mathrm{d}s \\ &= \int_{s=0}^{s=t} \exp(2\alpha s)\,\mathrm{d}s \\ &= \frac{\exp(2\alpha s)}{2\alpha}\Bigg]_{s=0}^{s=t} \\ &= \frac{\exp(2\alpha t) - 1}{2\alpha} \end{aligned}$$

Substituting the above result into Eq. (2.10.10)

$$\mathrm{Var}[X_t] = \sigma^2 \exp(-2\alpha t)\left\{\frac{\exp(2\alpha t) - 1}{2\alpha}\right\}$$

which yields the following expression for the variance

$$\mathrm{Var}[X_t] = \sigma^2\left\{\frac{1 - \exp(-2\alpha t)}{2\alpha}\right\} \tag{2.10.11}$$

The expressions for the mean and variance derived in Eqs. (2.10.7) and (2.10.11) allow us to write the distribution of X_t as

$$X_t \sim \mathrm{N}\left(X_{t_0} \exp(-\alpha t), \sigma^2\left\{\frac{1 - \exp(-2\alpha t)}{2\alpha}\right\}\right) \tag{2.10.12}$$

which, if $X_{t_0} = 0$, reduces to

$$X_t \sim \mathrm{N}\left(0, \sigma^2\left\{\frac{1 - \exp(-2\alpha t)}{2\alpha}\right\}\right) \tag{2.10.13}$$

The transition density from X_{t_0} to X_t is:

$$P(X_t | X_{t_0}) = \sqrt{\frac{K}{2\pi(1-\gamma^2)}} \exp\left\{-\frac{K(X_t - X_{t_0}\exp(-\alpha(t-t_0)))^2}{1-\gamma^2}\right\} \tag{2.10.14}$$

where $K = 2\alpha/\sigma^2$ and $\gamma = \exp(-\alpha(t - t_0))$.

Ornstein–Uhlenbeck stochastic paths can thus be simulated using

$$X_{t+dt} = X_t \exp(-\alpha\, \mathrm{d}t) + \sigma\sqrt{\frac{1-\exp(-2\alpha\, \mathrm{d}t)}{2\alpha}}\, \mathrm{d}Z \tag{2.10.15}$$

From Eq. (2.10.14) we can write

$$E[X_{t+\mathrm{d}t}|X_t] = X_t \exp(-\alpha\, \mathrm{d}t)$$

$$\mathrm{Var}[X_{t+\mathrm{d}t}|X_t] = \sigma^2\left\{\frac{1-\exp(-2\alpha\, \mathrm{d}t)}{2\alpha}\right\}$$

We will now show that in the limit $\mathrm{d}t \to 0$, Eq. (2.10.15) reduces to the Ornstein–Uhlenbeck process given in Eq. (2.10.1).

For small $\mathrm{d}t$ we can take a first-order expansion of the exponentials in Eq. (2.10.15) to obtain

$$X_{t+\mathrm{d}t} = X_t\big\{(1-\alpha\, \mathrm{d}t)\big\} + \sigma\sqrt{\frac{(1-(1-2\alpha\, \mathrm{d}t))}{2\alpha}}\, \mathrm{d}Z$$

so

$$X_{t+\mathrm{d}t} = X_t - X_t\alpha\, \mathrm{d}t + \sigma\sqrt{\frac{2\alpha\, \mathrm{d}t}{2\alpha}}\, \mathrm{d}Z$$

Therefore

$$X_{t+\mathrm{d}t} - X_t = -\alpha X_t\, \mathrm{d}t + \sigma\sqrt{\mathrm{d}t}\, \mathrm{d}Z$$

which is

$$\mathrm{d}X_t = -\alpha X_t\, \mathrm{d}t + \sigma\, \mathrm{d}W_t$$

2.11 The Ornstein–Uhlenbeck bridge

Let an Ornstein–Uhlenbeck process have value X_{t_0} at time t_0 and X_{t_1} at time t_1. We are interested in the distribution of X_t at an intermediate point, that is $P(X_t|\{X_{t_0}, X_{t_1}\})$, where $t_0 < t < t_1$.

We will show that X_t is a Gaussian with conditional mean

$$\mu_t = X_{t_0}\exp\big(-\alpha(t-t_0)\big)\left\{\frac{1-\exp(-2\alpha(t_1-t))}{1-\gamma^2}\right\} + X_{t_1}\exp\big(-\alpha(t_1-t)\big)\left\{\frac{1-\exp(-2\alpha(t-t_0))}{1-\gamma^2}\right\} \tag{2.11.1}$$

and conditional variance

$$V_t = \frac{(1-\exp(-2\alpha(t-t_0)))(1-\exp(-2\alpha(t_1-t)))}{2\alpha(1-\exp(-2\alpha(t_1-t_0)))} \tag{2.11.2}$$

where $\gamma = \exp(-\alpha(t_1-t_0))$.

PROOF. The standard Ornstein–Uhlenbeck process ($\sigma = 1$) is defined by the process:

$$\mathrm{d}X_t = -\alpha X_t \,\mathrm{d}t + \sqrt{\mathrm{d}t}\,\mathrm{d}Z_t \tag{2.11.3}$$

From Section 2.10 we have that

$$X_t = X_{t_0}\exp\big(-\alpha(t-t_0)\big) + \left\{\frac{1-\exp(-2\alpha(t-t_0))}{2\alpha}\right\}\mathrm{d}Z_t \tag{2.11.4}$$

and that the transition density from X_{t_0} to X_t is

$$P(X_t|X_{t_0}) = \frac{\sqrt{2\alpha}}{\sqrt{2\pi(1-\exp(-2\alpha(t-t_0)))}} \times \exp\left\{-\frac{\alpha(X_t - X_{t_0}\exp(-\alpha(t-t_0)))^2}{1-\exp(-2\alpha(t-t_0))}\right\} \tag{2.11.5}$$

The joint density of X_t and X_{t_1} given X_{t_0} is:

$$P\big(\{X_t, X_{t_1}\}|X_{t_0}\big) = P(X_{t_1}|X_t)P(X_t|X_{t_0}) \tag{2.11.6}$$

We thus have:

$$P\big(\{X_t, X_{t_1}\}|X_{t_0}\big) = \kappa\exp\left\{-\frac{\alpha(X_t - X_{t_0}\exp(-\alpha(t-t_0)))^2}{(1-\exp(-2\alpha(t-t_0)))}\right\} \times \exp\left\{-\frac{\alpha(X_{t_1} - X_t\exp(-\alpha(t_1-t)))^2}{1-\exp(-2\alpha(t_1-t))}\right\}$$

where

$$\kappa = \frac{1}{2\pi}\frac{2\alpha}{\sqrt{(1-\exp(-2\alpha(t-t_0)))(1-\exp(-2\alpha(t_1-t)))}}$$

The distribution of X_t given X_{t_0} and X_{t_1}, $P(X_t|X_{t_0}, X_{t_1})$ is:

$$P\big(X_t|\{X_{t_0}, X_{t_1}\}\big) = \frac{P(\{X_t, X_{t_1}\}|X_{t_0})}{P(X_{t_1}|X_{t_0})} \tag{2.11.7}$$

where

$$P(X_{t_1}|X_{t_0}) = \frac{\sqrt{2\alpha}}{\sqrt{2\pi(1-\exp(-2\alpha(t_1-t_0)))}} \times \exp\left\{-\frac{\alpha(X_{t_1} - X_{t_0}\exp(-\alpha(t_1-t_0)))^2}{1-\exp(-2\alpha(t_1-t_0))}\right\} \tag{2.11.8}$$

After some algebra we can re-express Eq. (2.11.7) as

$$P\big(X_t|\{X_{t_0}, X_{t_1}\}\big) = \sqrt{\frac{\alpha}{\pi\phi_t}}\exp\{A\} \tag{2.11.9}$$

where

$$A = -\frac{\alpha}{\phi}(B_1 + B_2 - B_3)$$

$$B_1 = \left\{X_t^2 + X_{t_0}^2 \exp\left(-2\alpha(t - t_0)\right) - 2X_t X_{t_0} \exp\left(-\alpha(t - t_0)\right)\right\} \times \left\{\frac{1 - \exp(-2\alpha(t_1 - t))}{1 - \gamma^2}\right\} \tag{2.11.10}$$

$$B_2 = \left\{X_{t_1}^2 + X_t^2 \exp\left(-2\alpha(t_1 - t)\right) - 2X_t X_{t_1} \exp\left(-\alpha(t_1 - t)\right)\right\} \times \left\{\frac{1 - \exp(-2\alpha(t - t_0))}{1 - \gamma^2}\right\} \tag{2.11.11}$$

$$B_3 = \left\{X_{t_1}^2 + X_{t_0}^2 \exp\left(-2\alpha(t_1 - t_0)\right) - 2X_{t_1} X_{t_0} \exp\left(-\alpha(t_1 - t_0)\right)\right\} \times \left\{1 - \exp\left(-2\alpha(t - t_0)\right)\right\}\left\{\frac{1 - \exp(-2\alpha(t_1 - t))}{(1 - \gamma^2)^2}\right\} \tag{2.11.12}$$

$$\phi_t = \frac{(1 - \exp(-2\alpha(t - t_0)))(1 - \exp(-2\alpha(t_1 - t)))}{1 - \exp(-2\alpha(t_1 - t_0))}$$

and

$$\gamma = \exp\left(-\alpha(t_1 - t_0)\right)$$

Let us now assume that $P(X_t|\{X_{t_0}, X_{t_1}\})$ is a normal distribution with conditional mean μ_t and conditional variance V_t. We thus have:

$$P\left(X_t|\{X_{t_0}, X_{t_1}\}\right) = \frac{1}{\sqrt{2\pi V_t}} \exp\left\{-\frac{(X_t - \mu_t)^2}{2V_t}\right\} \tag{2.11.13}$$

Equating Eqs. (2.11.7) and (2.11.13)

$$-\frac{1}{2V_t}\left\{(X_t - \mu_t)^2\right\} = -\frac{1}{2V_t}\left\{X_t^2 - \mu_t^2 - 2X_t\mu_t\right\} = -\frac{\alpha}{\phi}(B_1 + B_2 - B_3) \tag{2.11.14}$$

The conditional variance V_t can be obtained by noting that:

$$\frac{1}{2V_t} = \frac{\alpha}{\phi_t}$$

and hence:

$$V_t = \frac{\phi}{2\alpha} \tag{2.11.15}$$

so substituting for ϕ in Eq. (2.11.15) we obtain the following expression for the conditional variance:

$$V_t = \frac{(1 - \exp(-2\alpha(t - t_0)))(1 - \exp(-2\alpha(t_1 - t)))}{1 - \exp(-2\alpha(t_1 - t_0))} = \frac{\phi_t}{2\alpha}$$

The conditional mean can be obtained by noting that X_{t_0} and X_{t_1} are constants and the coefficients of X_t and X_t^2 in Eq. (2.11.14) must be the same. Comparing coefficients of X_t we thus have:

$$-2\mu_t = -2X_{t_0}\exp\big(-\alpha(t-t_0)\big)\left\{\frac{1-\exp(-2\alpha(t_1-t))}{1-\gamma^2}\right\} - 2X_{t_1}\exp\big(-\alpha(t_1-t)\big)\left\{\frac{1-\exp(-2\alpha(t-t_0))}{1-\gamma^2}\right\}$$

So the conditional mean μ_t is:

$$\mu_t = X_{t_0}\exp\big(-\alpha(t-t_0)\big)\left\{\frac{1-\exp(-2\alpha(t_1-t))}{1-\gamma^2}\right\} + X_{t_1}\exp\big(-\alpha(t_1-t)\big)\left\{\frac{1-\exp(-2\alpha(t-t_0))}{1-\gamma^2}\right\}$$

This completes the proof. □

Relation to the Brownian bridge

We will now prove that in the limit $(t_1 - t_0) \to 0$ the Brownian bridge result is obtained.

For the conditional mean, we have:

$$\mu = X_{t_0}\exp\big(-\alpha(t-t_0)\big)\left\{\frac{1-\exp(-2\alpha(t_1-t))}{1-\gamma^2}\right\} + X_{t_1}\exp\big(-\alpha(t_1-t)\big)\left\{\frac{1-\exp(-2\alpha(t-t_0))}{1-\gamma^2}\right\}$$

where:

$$\gamma = \exp\big(-\alpha(t_1-t_0)\big)$$

which is:

$$\mu = X_{t_0}\left\{\frac{\exp(-\alpha(t-t_0)) - \exp(-2\alpha(t_1-t))}{1-\exp(-2\alpha(t_1-t_0)-\alpha(t-t_0))}\right\} + X_{t_1}\left\{\frac{\exp(-\alpha(t_1-t)) - \exp(-2\alpha(t-t_0)-\alpha(t_1-t))}{1-\exp(-2\alpha(t_1-t_0))}\right\}$$

For small $t_1 - t_0$ both $t_1 - t$ and $t - t_0$ are small, so:

$$\mu \to X_{t_0}\left\{\frac{1-\alpha(t-t_0)-\{1-2\alpha(t_1-t)-\alpha(t-t_0)\}}{1-\{1-2\alpha(t_1-t_0)\}}\right\} + X_{t_1}\left\{\frac{1-\alpha(t_1-t)-\{1-2\alpha(t-t_0)-\alpha(t_1-t)\}}{1-\{1-2\alpha(t_1-t_0)\}}\right\}$$

which yields the Brownian bridge result for the conditional mean:

$$\mu \to X_{t_0}\frac{t_1-t_0}{t_1-t_0} + X_{t_1}\frac{t-t_0}{t_1-t_0}$$

For the conditional variance:

$$V_t = \frac{(1-\exp(-2\alpha(t-t_0)))(1-\exp(-2\alpha(t_1-t)))}{2\alpha(1-\exp(-2\alpha(t_1-t_0)))}$$

For small $t_1 - t_0$ both $t_1 - t$ and $t - t_0$ are small, so we can write:

$$V \to \frac{(1-\{1-2\alpha(t-t_0)\})(1-\{1-2\alpha(t_1-t)\})}{2\alpha(1-\{1-2\alpha(t_1-t_0)\})}$$

which yields the Brownian bridge result for the conditional variance:

$$V_t \to \frac{(t-t_0)(t_1-t)}{t_1-t_0}$$

2.12 Other useful results

2.12.1 Fubini's theorem

Fubini's theorem states that (for well-behaved functions) the value of a multidimensional integral is independent of the order in which the integral is evaluated.

For example, the two-dimensional integral of the function $f(X, Y)$ can be evaluated as:

$$\begin{aligned}\int_{X=a}^{b}\int_{Y=c}^{d} f(X,Y)\,\mathrm{d}X\,\mathrm{d}Y &= \int_{Y=c}^{d}\left\{\int_{X=a}^{b} f(X,Y)\,\mathrm{d}X\right\}\mathrm{d}Y \\ &= \int_{X=a}^{b}\left\{\int_{Y=c}^{d} f(X,Y)\,\mathrm{d}Y\right\}\mathrm{d}X\end{aligned}$$

We will mainly use this result in the form:

$$E\left[\int_{s=0}^{t} f(W,s)\,\mathrm{d}s\right] = \int_{s=0}^{t} E\big[f(W,s)\big]\,\mathrm{d}s \tag{2.12.1}$$

Since

$$E\big[f(W,s)\big] = \int_{-\infty}^{\infty} P(W,s)\,f(W,s)\,\mathrm{d}W$$

where $P(W, s)$ is the probability density function of $f(W, s)$, we can thus write Eq. (2.12.1) in full as:

$$\begin{aligned}&\int_{W=-\infty}^{\infty}\left\{\int_{s=0}^{t} P(W,s)\,f(W,s)\,\mathrm{d}s\right\}\mathrm{d}W \\ &\quad= \int_{s=0}^{t}\left\{\int_{W=-\infty}^{\infty} P(W,s)\,f(W,s)\,\mathrm{d}W\right\}\mathrm{d}s\end{aligned}$$

2.12.2 Ito's isometry

The expected value of the integral of the well-behaved function $f(W_t, t)$ satisfies:

$$E\left[\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)^2\right] = E\left[\int_{s=t_a}^{t_b} \{f(W_s, s)\}^2\,\mathrm{d}s\right] \tag{2.12.2}$$

Proof. We first use the following approximation:

$$\int_{s=t_a}^{s=t_b} f(W, s)\,\mathrm{d}W_s = \sum_{i=0}^{n-1} f(W_{t_i}, t_i)\{W_{t_{i+1}} - W_{t_i}\}$$

where $a < t_0 < t_1 < \cdots < t_n < t_b$ and $t_{i+1} - t_i = \mathrm{d}t$. Thus the integral on the left-hand side of Eq. (2.12.2) is:

$$\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)^2 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} f(W_{t_i}, t_i) f(W_{t_j}, t_j)\{W_{t_{i+1}} - W_{t_i}\}\{W_{t_{j+1}} - W_{t_j}\} \tag{2.12.3}$$

Taking expectations of Eq. (2.12.3) we obtain:

$$E\left[\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)\right]^2 = E\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} f(W_{t_i}, t_i) f(W_{t_j}, t_j)\{W_{t_{i+1}} - W_{t_i}\}\{W_{t_{j+1}} - W_{t_j}\}\right] \tag{2.12.4}$$

which means that:

$$E\left[\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)\right]^2 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} f(W_{t_i}, t_i) f(W_{t_j}, t_j) E\left[\{W_{t_{i+1}} - W_{t_i}\}\{W_{t_{j+1}} - W_{t_j}\}\right] \tag{2.12.5}$$

However from the Brownian motion property (iii) in Section 2.1 we have:

$$E\left[\{W_{t_{i+1}} - W_{t_i}\}\{W_{t_{j+1}} - W_{t_j}\}\right] = 0 \quad \text{when } i \neq j \text{ and } \mathrm{d}t \text{ when } i = j$$

Therefore Eq. (2.12.5) can be rewritten as:

$$E\left[\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)\right]^2 = \sum_{i=0}^{n-1}\{f(W_{t_i}, t_i)\}^2\,\mathrm{d}t$$

which means:

$$E\left[\left(\int_{s=t_a}^{t_b} f(W_s, s)\,\mathrm{d}W_s\right)\right]^2 = E\left[\int_{s=t_a}^{t_b} \{f(W_s, s)\}^2\,\mathrm{d}s\right]$$

□

2.12.3 Expectation of a stochastic integral

If $f(t)$ is a deterministic function of time then

$$E\left[\int_{s=a}^{s=b} f(s)\,\mathrm{d}W_s\right] = 0 \tag{2.12.6}$$

Proof. We first express the integral (2.12.6) by the following summation:

$$\int_{s=t_a}^{s=t_b} f(s)\,\mathrm{d}W_s = \sum_{i=0}^{n-1} f(t_i)\{W_{t_{i+1}} - W_{t_i}\},$$

where $a < t_0 < t_1 < \cdots < t_n < t_b$

Taking expectations of the above equation yields:

$$\begin{aligned} E\left[\int_{s=a}^{s=b} f(s)\,\mathrm{d}W_s\right] &= E\left[\sum_{i=0}^{n-1} f(t_i)\{W_{t_{i+1}} - W_{t_i}\}\right] \\ &= \sum_{i=0}^{n-1} f(t_i)E[W_{t_{i+1}} - W_{t_i}] \\ &= 0 \end{aligned}$$

where we have used $E[W_{t_{i+1}} - W_{t_i}] = 0$, which is Brownian motion property (iii) in Section 2.1. □

2.13 Selected problems

In this section we provide various problems that test the reader's understanding of stochastic calculus. The answers are given in the appendix at the end of the book.

Problem 1 (Problem 4.5, Øksendal (2003)). Let $\beta_t^k = E[W_t^k], k = 0, 1, 2, \ldots,$ $t \geqslant 0$, where $W_{t_0} = 0$.

(a) Show using Ito's formula for $k = 2, 3, 4, \ldots,$ that:

$$\beta_t^k = \frac{1}{2}(k-1)\int_{s=0}^{t} \beta_s^{k-2}\,\mathrm{d}s$$

(b) Deduce that $E[W_t^4] = 3t^2$
(c) What is $E[W_t^6]$?

Problem 2 (Problem 5.4(ii), Øksendal (2003)). Solve the stochastic differential equation:

$$\mathrm{d}X_t = X_t\,\mathrm{d}t + \mathrm{d}W_t$$

PROBLEM 3 (Problem 5.4(iii), Øksendal (2003)). Solve the stochastic differential equation:

$$\mathrm{d}X_t = -X_t\,\mathrm{d}t + \exp(-t)\,\mathrm{d}W_t$$

PROBLEM 4 (Problem 4.2, Øksendal (2003)). Use Ito's formula to prove that

$$\int_{s=0} W_s^2\,\mathrm{d}W_s = \frac{1}{3}W_t^3 - \int_{s=0}^{t} W_s\,\mathrm{d}s$$

where $W_{t_0} = 0$.

PROBLEM 5 (Problem 5.6, Øksendal (2003)). Solve:

$$\mathrm{d}Y_t = r\,\mathrm{d}t + \alpha Y_t\,\mathrm{d}W_t$$

where r and α are real constants. Use the integrating factor $F_t = \exp(-\alpha W_t + (\alpha^2/2)t)$.

PROBLEM 6 (Problem 5.7, Øksendal (2003)). The mean reverting Ornstein–Uhlenbeck process is the solution X_t of the stochastic differential equation:

$$\mathrm{d}X_t = (m - X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$$

where m and σ are constants.

(a) Solve this equation
(b) Find $E[X_t]$ and $\mathrm{Var}[X_t] = E[\{X_t - E[X_t]\}^2]$.

PROBLEM 7. Consider the equation $\mathrm{d}S_t = \mu_t S_t\,\mathrm{d}t + \sigma_t S_t\,\mathrm{d}W_t$ where the value of S_t at time $t = 0$ is denoted by S_0.

(a) Show that the mean is:

$$E\big[\log(S_t)\big] = \log(S_0) + \int_{\tau=0}^{t}\left\{\mu_\tau - \frac{\sigma_\tau^2}{2}\right\}\mathrm{d}\tau$$

(b) Show that the variance is:

$$\mathrm{Var}\big[\log(S_t)\big] = \int_{\tau=0}^{t}\sigma_\tau^2\,\mathrm{d}\tau$$

PROBLEM 8. Prove that if $\phi = \exp(tW_t)$ then

$$\mathrm{d}\phi = \phi\left(W_t + \frac{t^2}{2}\right)\mathrm{d}t + t\phi\,\mathrm{d}W_t$$

PROBLEM 9 (Problem 4.4, Øksendal (2003)). Define:

$$Z_t = \exp\left(\int_{s=0}^{t}\theta_s\,\mathrm{d}W_s - \frac{1}{2}\int_{s=0}^{t}\theta_s^2\,\mathrm{d}s\right)$$

Use Ito's formula to prove that

$$\mathrm{d}Z_t = Z_t\theta_t\,\mathrm{d}W_t$$

PROBLEM 10. Let $S_t = S_0 \exp(\mu t + \sigma W_t)$ where μ and σ are constants.

(a) Show by Ito's lemma that:

$$\mathrm{d}S_t = \left(\mu + \frac{\sigma^2}{2}\right) S_t \,\mathrm{d}t + \sigma S_t \,\mathrm{d}W_t$$

(b) Show that:

$$E[S_t] - E[S_0] = \left(\mu + \frac{\sigma^2}{2}\right) \int_{\tau=0}^{t} E\big[S(\tau)\big] \,\mathrm{d}\tau$$

(c) Show that:

$$E[S_t] = S_0 \exp\left(\mu t + \frac{\sigma^2}{2} t\right)$$

PROBLEM 11 (Problem 4.3, Øksendal (2003)). Let X_t, Y_t be stochastic processes. Prove that:

$$\mathrm{d}(X_t Y_t) = X_t \,\mathrm{d}Y_t + Y_t \,\mathrm{d}X_t + E[\mathrm{d}X_t \,\mathrm{d}Y_t]$$

Deduce the following general integration by parts formula:

$$\int_{s=0}^{t} X_s \,\mathrm{d}Y_s = X_t Y_t - X_{t_0} Y_{t_0} - \int_{s=0}^{t} Y_s \,\mathrm{d}X_s - \int_{s=0}^{t} E[\mathrm{d}X_s \,\mathrm{d}Y_s]$$

This page intentionally left blank

3 Generation of random variates

3.1 Introduction

Monte Carlo simulation and random number generation are techniques that are widely used in financial engineering as a means of assessing the level of exposure to risk. Typical applications include the pricing of financial derivatives and scenario generation in portfolio management. In fact many of the financial applications that use Monte Carlo simulation involve the evaluation of various stochastic integrals which are related to the probabilities of particular events occurring.

In many cases, however, the assumptions of constant volatility and a lognormal distribution for S_T are quite restrictive. Real financial applications may require a variety of extensions to the standard Black–Scholes model. Common requirements are for: nonlognormal distributions, time-varying volatilities, caps, floors, barriers, etc. In these circumstances, it is often the case that there is no closed form solution to the problem. Monte Carlo simulation can then provide a very useful means of evaluating the required integrals.

When we evaluate the integral of a function, $f(x)$, in the s-dimensional unit cube, I^S, by the Monte Carlo method we are in fact calculating the average of the function at a set of randomly sampled points. This means that each point adds linearly to the accumulated sum that will become the integral and also linearly to the accumulated sum of squares that will become the variance of the integral.

When there are N sample points, the integral is:

$$v = \frac{1}{N}\sum_{i=1}^{N} f\left(x^i\right)$$

where v is used to denote the approximation to the integral and $x^1, x^2, \ldots, x^N$ are the N, s-dimensional, sample points. If a pseudo-random number generator is used the points x^i will be (*should be*) independently and identically distributed. From standard statistical results we can then estimate the expected error of the integral as follows:

If we set $\chi^i = f(x^i)$ then since x^i is independently and identically distributed χ^i is also independently and identically distributed. The mean of χ^i is v and we

will denote the variance as $\text{Var}[\chi^i] = \Delta^2$. It is a well-known statistical property that the variance of v is given by $\text{Var}[v] = N^{-1}\Delta^2$ (see Appendix E.1 for further details). We can therefore conclude that the estimated integral v has a standard error of $N^{-1/2}\Delta$. This means that the estimated error of the integral will decrease at the rate of $N^{-1/2}$.

It is possible to achieve faster convergence than this if the sample points are chosen to lie on a Cartesian grid. If we sample each grid point exactly once, then the Monte Carlo method effectively becomes a deterministic quadrature scheme, whose fractional error decreases at the rate of N^{-1} or faster. The trouble with the grid approach is that it is necessary to decide in advance how fine it should be, and all the grid points need to be used. It is therefore not possible to sample until some convergence criterion has been met.

Quasi-random number sequences seek to bridge the gap between the flexibility of pseudo-random number generators and the advantages of a regular grid. They are designed to have a high level of uniformity in multidimensional space, but unlike pseudo-random numbers they are not statistically independent.

3.2 Pseudo-random and quasi-random sequences

Here we consider the generation of multidimensional pseudo-random and quasi-random sequences to approximate the multidimensional uniform distribution over the interval [0, 1], that is the distribution U(0, 1).

Quasi-random numbers are also called low-discrepancy sequences. The discrepancy of a sequence is a measure of its uniformity and is defined as follows:

Given a set of points $x^1, x^2, \ldots, x^N \in I^S$ and a subset $G \subset I^S$, define the counting function $S_N(G)$ as the number of points $x^i \in G$. For each $x = (x_1, x_2, \ldots, x_s) \in I^S$, let G_x be the rectangular s-dimensional region $G_x = [0, x_1) \times [0, x_2) \times \cdots \times [0, x_s)$, with volume $x_1, x_2, \ldots, x_n$. Then the discrepancy of the points $x^1, x^2, \ldots, x^N$ is given by:

$$D_N^*\left(x^1, x^2, \ldots, x^N\right) = \sup_{x \in I^S} \left| S_N(G_x) - N x_1 x_2, \ldots, x_s \right|$$

The discrepancy is therefore computed by comparing the actual number of sample points in a given volume of multidimensional space with the number of sample points that should be there assuming a uniform distribution.

It can be shown that the discrepancy of the first terms of quasi-random sequence has the form:

$$D_N^*\left(x^1, x^2, \ldots, x^N\right) \leqslant C_S(\log N)^S + \mathrm{O}\left((\log N)^{S-1}\right)$$

for all $N \geqslant 2$.

The principal aim in the construction of low-discrepancy sequences is thus to find sequences in which the constant is as small as possible. Various sequences have been constructed to achieve this goal. Here we consider the following quasi-random sequences: Niederreiter, Sobol, and Faure.

The results of using various random number generators are shown below. Figures 3.1–3.3 illustrate the visual uniformity of the sequences. They were cre-

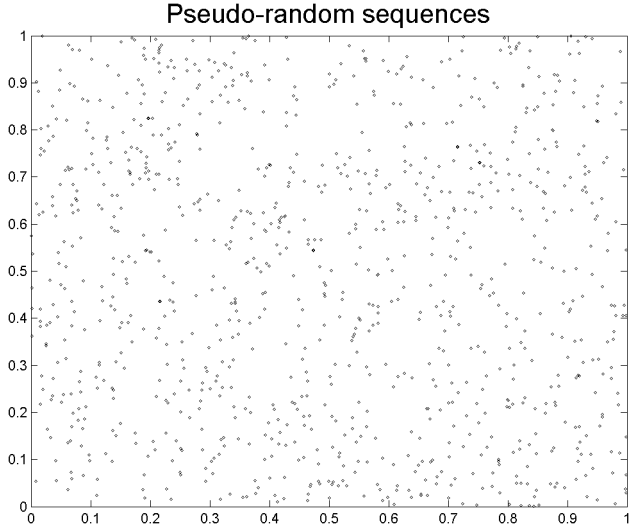

Figure 3.1 The scatter diagram formed by one thousand points from a 16-dimensional U(0, 1) pseudo-random sequence. For each point the 4th-dimensional component is plotted against the 5th-dimensional component.

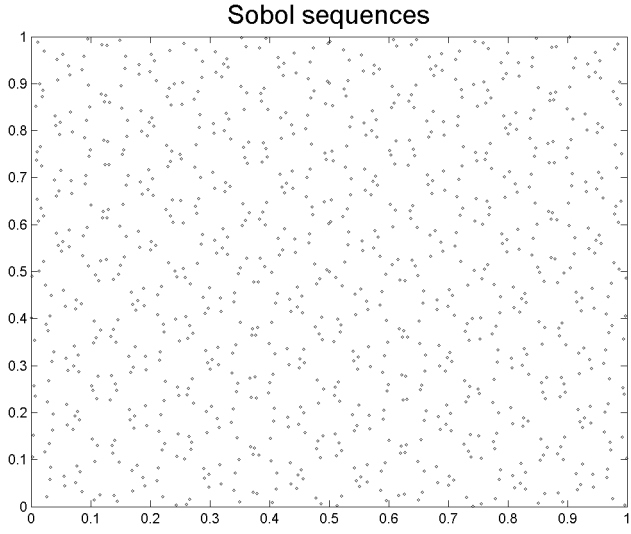

Figure 3.2 The scatter diagram formed by one thousand points from a 16-dimensional U(0, 1) Sobol sequence. For each point the 4th-dimensional component is plotted against the 5th-dimensional component.

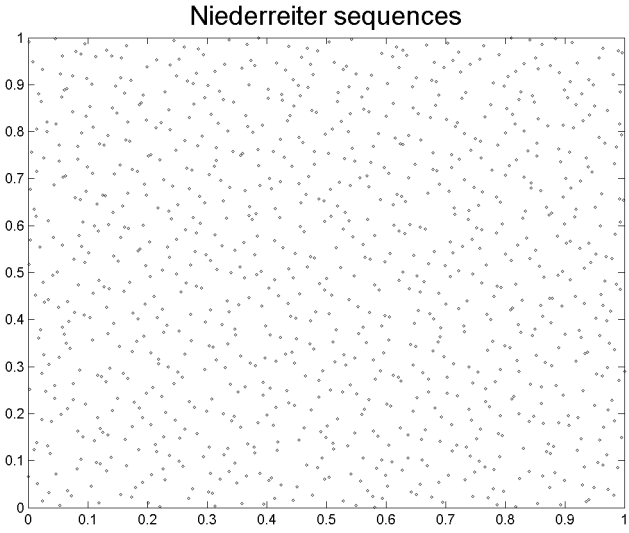

Figure 3.3 The scatter diagram formed by one thousand points from a 16-dimensional U(0, 1) Niederreiter sequence. For each point the 4th-dimensional component is plotted against the 5th-dimensional component.

ated by generating one thousand 16-dimensional U(0, 1) sample points, and then plotting the 4th-dimensional component of each point against its 5th-dimensional component.

In Fig. 3.1, it can be seen that the pseudo-random sequence exhibits clustering of points, and there are regions with no points at all.

Visual inspection of Figs. 3.2 and 3.3 shows that both the Sobol and Niederreiter quasi-random sequences appear to cover the area more uniformly.

It is interesting to note that the Sobol sequence appears to be a structured lattice which still has some gaps. The Niederreiter sequence, on the other hand, appears to be more irregular and covers the area better. However, we cannot automatically conclude from this that the Niederreiter sequence is the best. This is because we have not considered all the other possible pairs of dimensions.

Perhaps the easiest way to evaluate the random number sequences is to use them to calculate an integral.

In Fig. 3.4 Monte Carlo results are presented for the calculation of the 6-dimensional integral:

$$I = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \prod_{i=1}^{6} \cos(i x_i)\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3\, \mathrm{d}x_4\, \mathrm{d}x_5\, \mathrm{d}x_6$$

The exact value of this integral is:

$$I = \prod_{i=1}^{6} \sin(i)$$

which for $i = 6$, gives $I = -0.0219$.

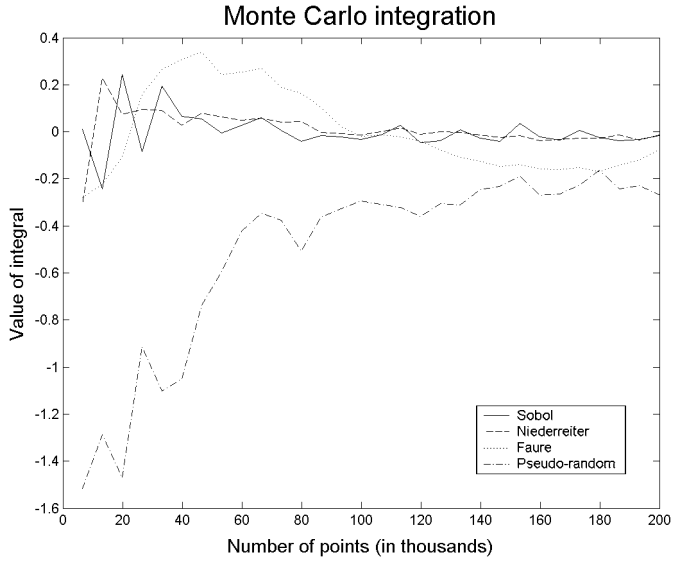

Figure 3.4 Monte Carlo integration using random numbers.

It can be seen that the pseudo-random sequence gives the worst performance. But as the number of points increases, its approximation to the integral improves. Of the quasi-random sequences, it can be seen that the Faure sequence has the worst performance, while both the Sobol and Neiderreiter sequences give rapid convergence to the solution.

Finance literature contains many references to the benefits of using quasi-random numbers for computing important financial integrals. For instance, Brotherton-Ratcliffe (1994) discusses the use of Sobol sequences for the valuation of geometric mean stock options, and provides results that show that the root-mean-squared pricing error obtained using quasi-random numbers is considerably less than that computed with pseudo-random numbers. Another financial application of quasi-random numbers is the efficient pricing mortgage backed securities, Caflisch, Morokoff, and Owen (1997). Here Brownian bridge techniques are employed to reduce the effective dimension of the problem and thus provide greater pricing accuracy than if pseudo-random numbers were used.

3.3 Generation of multivariate distributions: independent variates

In this section we show how to generate multivariate distributions that contain independent variates; that is, the variates have zero correlation.

3.3.1 Normal distribution

The most fundamental distribution is the univariate standard normal distribution, N(0, 1), with zero mean and unit variance. In the case of p independent variates this takes the form of a p variate independent normal distribution N(0, I_p) with zero mean and $p \times p$ unit covariance matrix I_p.

First we will quote a result concerning multivariate probability density functions; see Press, Teukolsky, Vetterling, and Flannery (1992). If $x_1, x_2, \ldots$ are random variates with a joint probability density function $p(x_1, x_2, \ldots)$, and if there are an equal number of y variates $y_1, y_2, \ldots$ that are functions of the x's, then the joint probability density function of the y variates, $p(y_1, y_2, \ldots)$, is given by the following expression:

$$p(y_1, y_2, \ldots)\,\mathrm{d}y_1\,\mathrm{d}y_2, \ldots = p(x_1, x_2, \ldots)\mathcal{J}_{x,y}\,\mathrm{d}y_1\,\mathrm{d}y_2, \ldots \tag{3.3.1}$$

where $\mathcal{J}_{x,y}$ is the Jacobian determinant of the x's with respect to the y's.

An important application of this result is the Box–Muller transformation in which a p variate independent normal distribution N(0, I_p) is generated from a p variate uniform distribution U(0, 1); see Box and Muller (1958).

We will now describe how the method works.

Consider two independently distributed N(0, 1) variables x and y, and use the polar transformation to obtain:

$$x = r\cos\theta, \qquad y = r\sin\theta, \quad \text{and} \quad r^2 = x^2 + y^2 \tag{3.3.2}$$

From Eq. (3.3.1) the joint probability density functions $f(r, \theta)$ and $f(x, y)$ obey the equation

$$f(r, \theta)\,\mathrm{d}r\,\mathrm{d}\theta = f(x, y)\mathcal{J}_{xy,r\theta}\,\mathrm{d}r\,\mathrm{d}\theta$$

where the Jacobian is

$$\mathcal{J}_{xy,r\theta} = \begin{vmatrix} \cos\theta & \sin\theta \\ -r\sin\theta & r\cos\theta \end{vmatrix} = r$$

We therefore have

$$f(r, \theta) = rf(x, y) \tag{3.3.3}$$

Furthermore since x and y are independent N(0, 1)

$$f(x, y) = f(x)f(y)$$

where

$$f(x) = \frac{\mathrm{e}^{-x^2/2}}{\sqrt{2\pi}} \quad \text{and} \quad f(y) = \frac{\mathrm{e}^{-y^2/2}}{\sqrt{2\pi}}$$

Therefore:

$$f(r, \theta) = rf(x)f(y) = r\frac{\mathrm{e}^{-x^2/2}}{\sqrt{2\pi}}\frac{\mathrm{e}^{-y^2/2}}{\sqrt{2\pi}}$$

which gives

$$f(r,\theta) = \frac{r}{2\pi}\mathrm{e}^{(-x^2+y^2)/2} = \frac{1}{2\pi} r\mathrm{e}^{-r^2/2} = f(\theta)f(r) \tag{3.3.4}$$

where $f(\theta) = 1/(2\pi)$, $f(r) = r\mathrm{e}^{-r^2/2}$ are independent probability density functions.

The corresponding cumulative probability distribution functions $F(\theta)$ and $F(r)$ can be found by evaluating the following integrals:

$$F(\theta) = \frac{1}{2\pi}\int_0^\theta \mathrm{d}\theta = \frac{\theta}{2\pi}$$

and

$$F(r) = \int_0^r r\mathrm{e}^{-r^2/2}\,\mathrm{d}r = \left[-\mathrm{e}^{-r^2/2}\right]_0^r = 1 - \mathrm{e}^{-r^2/2}$$

We now want to draw variates $\hat{r}$ and $\hat{\theta}$ from the probability distributions $f(r)$ and $f(\theta)$ respectively. To do this we will use the result (see for example Evans, Hastings, and Peacock (2000)), that a uniform variate $\bar{u}$, from the distribution U(0, 1) can be transformed into a variate $\bar{v}$ from the distribution $f(v)$ by using $\bar{v} = F^{-1}(\bar{u})$, or equivalently $F(\bar{v}) = \bar{u}$. The variates $\bar{v}$ are thus found by first drawing the uniform variate $\bar{u}$ and then finding the value of v which makes cumulative distribution function $F(v)$ equal to $\bar{u}$.

Therefore, if variates V_1' and V_2' are from U(0, 1), then the variates $\hat{r}$ and $\hat{\theta}$ which satisfy $V_1' = F(\hat{r}) = 1 - \mathrm{e}^{-\hat{r}^2/2}$, and $V_2' = F(\hat{\theta}) = \hat{\theta}/(2\pi)$ are from the distributions $f(r)$ and $f(\theta)$ respectively.

For convenience we will define the U(0, 1) variates

$$V_1 = 1 - V_1' = \mathrm{e}^{-\hat{r}^2/2} \quad \text{and} \quad V_2 = V_2'$$

So we have:

$$V_1 = \mathrm{e}^{-\hat{r}^2/2}, \qquad V_2 = \frac{\hat{\theta}}{2\pi}$$

and

$$\log V_1 = \frac{-\hat{r}^2}{2}, \quad \hat{r} = (-2\log V_1)^{1/2}, \quad \text{and} \quad \hat{\theta} = 2\pi V_2$$

Since $\hat{r}$ is from the same distribution as r, and $\hat{\theta}$ is from the same distribution as θ, we can use

$$\log V_1 = -r^2/2, \quad r = (-2\log V_1)^{1/2}, \quad \text{and} \quad \theta = 2\pi V_2$$

Substituting these results into Eq. (3.3.2) gives

$$x = (-2\log V_1)^{1/2}\cos 2\pi V_2, \qquad y = (-2\log V_1)^{1/2}\sin 2\pi V_2 \tag{3.3.5}$$

where x and y are N(0, 1).

The Box–Muller method is contained in Eq. (3.3.5), which shows that the N(0, 1) variates are generated in pairs from the uniform U(0, 1) variates V_1 and V_2.

Since the N(0, 1) variates are created two at a time, if we want to generate a normal distribution with an odd number of dimensions, n_{odd}, it is necessary to generate $n_{odd} + 1$ dimensions and discard one of the dimensions.

It is easy to modify Eq. (3.3.5) so that we can specify the means and variances of the variates x and y; this is accomplished as follows:

$$\begin{aligned} x &= \sigma_1(-2\log V_1)^{1/2}\cos 2\pi V_2 + \mu_1, \\ y &= \sigma_2(-2\log V_1)^{1/2}\sin 2\pi V_2 + \mu_2 \end{aligned} \tag{3.3.6}$$

where the distributions of x and y are now

$$x \sim \mathrm{N}\left(\mu_1, \sigma_1^2\right) \quad \text{and} \quad y \sim \mathrm{N}\left(\mu_2, \sigma_2^2\right)$$

Code excerpt 3.1 illustrates how to generate quasi-random normal variates with given means and standard deviations.

```
long Quasi_Normal_Independent(long fcall, long seq, double xmean[], double std[],
                              long idim, double quasi[])
{
/* Input parameters:
   ================
   fcall    -  if fcall == 1 then it is an initialisation call,
               if fcall == 0 then a continuation call
   seq      -  if seq == 0 then a Faure sequence, if seq == 1 then a Niederreiter sequence,
               if seq == 2 then a Sobol sequence
   xmean[]  -  the means of the independent normal variates
   std[]    -  the standard deviations of the independent normal variates
   idim     -  the number of independent normal variates, idim must be less than 40

   Output parameters:
   ==================
   quasi[]  -  the elements quasi[0], .. quasi[idim-1] contain the independent normal variates
*/

long ierr, i, j;
double twopi, v1, v2, pi;
long ind1, ind2;
#define QUASI(I) quasi[(I)-1]
#define STD(I) std[(I)-1]
#define XMEAN(I) xmean[(I)-1]

    if ((idim / 2) * 2 != idim) {
        printf("Error on entry, idim is not an even number: idim = %ld\n" ,idim);
        return 1;
    } else if (idim > 40) {
        printf("On entry, idim > 40: idim = %ld\n" ,idim);
        return 1;
    }
    for (i = 1; i <= idim; ++i) {
      if (STD(i) <= 0.0) {
        printf("On entry, the standard deviation is not greater than zero:
                STD(%ld) = %12.4f\n" ,i,STD(i));
        return 1;
      }
    }
    pi = 4.0*atan(1.0);
```

Code excerpt 3.1 Generating quasi-random normal variates using the Box–Muller transformation.

```
    if (fcall) { /* first call for initialisation */
      if (seq == 0) {
        Generate_Faure_Sequence(fcall, idim, &QUASI(1));
      }
      else if (seq == 1) {
          Generate_Niederreiter_Sequence(fcall, idim, &QUASI(1));
      }
      else if (seq == 2) {
          Generate_Sobol_Sequence(fcall, idim, &QUASI(1));
      }
    } else { /* a continuation call */
       if (seq == 0) {
          Generate_Faure_Sequence(fcall, idim, &QUASI(1));
       }
       else if (seq == 1) {
          Generate_ Niederreiter_Sequence(fcall, idim, &QUASI(1));
       }
       else if (seq == 2) {
          Generate_Sobol_Sequence(fcall, idim, &QUASI(1));
       }
       for (i = 1; i <= idim/2; ++i) { /* generate the normal variates */
           ind1 = i * 2 - 1;
           ind2 = i * 2;
           twopi = pi * 2.0;
           v1 = sqrt(log(QUASI(ind1)) * -2.0);
           v2 = twopi * QUASI(ind2);
           QUASI(ind1) = XMEAN(ind1) + STD(ind1) * v1 * cos(v2);
           QUASI(ind2) = XMEAN(ind2) + STD(ind2) * v1 * sin(v2);
        }
     }
     return 0 ;
}
```

Code excerpt 3.1 (*Continued*).

3.3.2 Lognormal distribution

The lognormal distribution can be generated from the normal distribution discussed in the previous section by means of a simple transformation. If $y \sim \mathrm{N}(\mu, \sigma^2)$ and $y = \log(x)$ then $x = \exp(y)$, and we say that the variable x has the lognormal distribution $\Lambda(\mu, \sigma^2)$.

The lognormal density function is:

$$f(x) = \frac{1}{x\sigma(2\pi)^{1/2}} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right) \tag{3.3.7}$$

If $z_i, i = 1, \ldots, p$, are independent normal variates $\mathrm{N}(\mu_i, \sigma_i^2)$, $i = 1, \ldots, p$, then lognormal variates $\ell_i, i = 1, \ldots, p$, can be generated using the transformation:

$$\ell_i = \exp(z_i), \quad i = 1, \ldots, p, \tag{3.3.8}$$

where the mean of the ith lognormal variate is

$$E[x_i] = \bar{m}_i = \exp\left(\mu_i + \frac{\sigma_i^2}{2}\right) \tag{3.3.9}$$

and the variance is

$$\mathrm{Var}[x_i] = s_i^2 = \exp\left(2\mu_i + \sigma_i^2\right)\left(\exp\left(\sigma_i^2\right) - 1\right) \tag{3.3.10}$$

The ratio of variance to the mean squared is therefore

$$\frac{s_i^2}{\bar{m}_i^2} = \exp\left(\sigma_i^2\right) - 1 \tag{3.3.11}$$

or equivalently

$$\sigma_i^2 = \log\left(1 + \frac{s_i^2}{\bar{m}_i^2}\right) \tag{3.3.12}$$

A lognormal distribution consisting of p independent variates with means $\bar{m}_i, i = 1, \ldots, p$, and variances $s_i^2, i = 1, \ldots, p$, can thus be generated using the following procedure.

First, generate the p independent normal variates:

$$z_i \sim \mathrm{N}(\mu_i, \sigma_i^2), \quad i = 1, \ldots, p,$$

where

$$\mu_i = \log(\bar{m}_i) - \frac{\sigma_i^2}{2} \tag{3.3.13}$$

and

$$\sigma_i^2 = \log\left(1 + \frac{s_i^2}{\bar{m}_i^2}\right) \tag{3.3.14}$$

Then create the independent lognormal variates using

$$\ell_i = \exp(z_i), \quad i = 1, \ldots, p$$

3.3.3 Student's t-distribution

If $S_t(\mu, \nu)$ represents the Student's t-distribution with mean μ and number of degrees of freedom ν, then variates $X \sim S_t(0, \nu)$ can be generated as follows:

$$X \sim \frac{Z}{\sqrt{Y/\nu}} \tag{3.3.15}$$

where $Z \sim \mathrm{N}(0, 1)$, and $Y \sim \chi_\nu^2$. The variance of X is:

$$E[X^2] = \frac{\nu}{\nu - 2}$$

Variates X' from a Student's t-distribution having ν degrees of freedom with mean μ and variance s can be generated by modifying Eq. (3.3.15) as follows:

$$X' \sim \mu + \frac{s^{1/2}}{\sqrt{\nu/(\nu - 2)}} \frac{Z}{\sqrt{Y/\nu}} \tag{3.3.16}$$

The probability density function, $f(x)$, for X' is:

$$f(x) = \frac{\Gamma((\nu + 1)/2)(\nu - 2)^{-1/2} s^{-1/2}}{\pi^{1/2}\Gamma(\nu/2)} \left[1 + \frac{(x - \mu)^2}{s(\nu - 2)}\right]^{-(\nu+1)/2} \tag{3.3.17}$$

where $\nu > 2$.

3.4 Generation of multivariate distributions: correlated variates

In this section we will show how to generate variates from a multivariate distribution with a known mean and a given covariance or correlation matrix. The methods described for covariance matrices are also applicable to correlation matrices, although in this case the generated variates are normalized to have unit variance.

3.4.1 Estimation of correlation and covariance

Here we show how to obtain a valid correlation matrix C_r or covariance matrix C from historic market data.

Let $\widehat{X}$ be an n by p data matrix, with the entries in the ith row corresponding to the ith observation, and the jth column containing the values of the jth variable. If we create a new matrix X such that the entries of the jth column of X are $X_{i,j} = \widehat{X}_{i,j} - \mu_j$, $i = 1, \ldots, n$, where μ_j is the mean of the jth column of $\widehat{X}$, then the p by p matrix $C = X^{\mathrm{T}}X$ is the covariance matrix of $\widehat{X}$.

Further, if another matrix $\overline{X}$ is defined such that $\overline{X}_{i,j} = \frac{\widehat{X}_{i,j} - \mu_j}{\sigma_j}$, $i = 1, \ldots, n$, where μ_j is the mean of the jth column of $\widehat{X}$, and σ_j is the standard deviation of the jth column, then the p by p matrix $C_r = \overline{X}^{\mathrm{T}}\overline{X}$ is the correlation matrix of $\widehat{X}$.

Correlation matrix

Let us first consider the properties of a valid correlation matrix. They are:

- The matrix is symmetric with unit diagonal
- The matrix has to be positive definite—that is, all the eigenvalues need to be positive.

  We will now show that the p by p matrix C_r is a valid correlation matrix if it can be factored as $C_r = X^{\mathrm{T}}X$, where X is a nonsingular (that is, full rank) n by m data matrix.

  The proof is as follows:
  Since X is nonsingular we can perform the singular value decomposition:

$$X = UKV^{\mathrm{T}}$$

where U is an n by n unitary matrix, K is an n by p matrix containing the (nonzero) singular values $\sigma_i, i = 1, \ldots, p$, as the *diagonal* elements and zero elsewhere, and V is a p by p unitary matrix.

We thus have

$$\begin{aligned} X^{\mathrm{T}}X &= \left(UKV^{\mathrm{T}}\right)^{\mathrm{T}}UKV^{\mathrm{T}} \\ &= VK^{\mathrm{T}}U^{\mathrm{T}}UKV^{\mathrm{T}} \end{aligned}$$

$$= V^{\mathrm{T}} K^{\mathrm{T}} K V^{\mathrm{T}} \qquad \text{since } U^{\mathrm{T}} U = U U^{\mathrm{T}} = I_n$$
$$= V \Sigma V^{\mathrm{T}}$$

Therefore

$$C_r = X^{\mathrm{T}} X = V \Sigma V^{\mathrm{T}}$$

where Σ is the p by p diagonal matrix containing the eigenvalues of C_r, and V is the corresponding matrix of eigenvectors. Since the ith eigenvalue satisfies $\lambda_i = \sigma_i^2$ it can be seen that all the eigenvalues are positive, and thus C_r must be positive definite.

If C_r is positive definite, then we can perform the Cholesky decomposition $C_r = LL^{\mathrm{T}}$ where L is a lower triangular matrix.

3.4.2 Repairing correlation and covariance matrices

There are situations when a supplied *correlation* matrix is not positive definite. Some of the reasons for this are:

- There may be missing data, or asynchronous data feeds. As a consequence the elements in the correlation matrix may have then been computed using pairwise correlations, with a variety of sequence lengths. Under these circumstances the equation $C_r = X^{\mathrm{T}} X$ is no longer true, and so C_r cannot be guaranteed to be positive definite
- Manual adjustment of a correlation matrix may have occurred to reflect expected market conditions. This especially occurs when the market crashes and certain stock prices become highly correlated
- There may be rounding error in computing $C_r = X^{\mathrm{T}} X$.

Under these circumstances the best that can be done is to try and repair the correlation matrix C_r into a valid correlation matrix $\widehat{C}_r$.

We proceed as follows.

When C_r is not positive definite (the Cholesky decomposition fails) then we use the eigen decomposition:

$$C_r = V \Sigma V^{\mathrm{T}}$$

where

$$\Sigma = \begin{pmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \lambda_p \end{pmatrix}$$

We then form the matrix

$$C_r^+ = V K K^{\mathrm{T}} V^{\mathrm{T}} = V K (V K)^{\mathrm{T}}$$

where the matrix K is formed by taking the square root of the maximum of each eigenvalue and a very small number ε (say $\sim 10^{-16}$). Thus:

$$K = \begin{pmatrix} \sqrt{\max(\lambda_1, \varepsilon)} & & & \\ & \sqrt{\max(\lambda_2, \varepsilon)} & & \\ & & \ddots & \\ & & & \sqrt{\max(\lambda_p, \varepsilon)} \end{pmatrix}$$

The matrix C_r^+ is not acceptable as a correlation matrix because, although real, symmetric and positive definite, its diagonal elements are not unity. It is possible to remedy this by premultiplying and postmultiplying C^+ by the diagonal matrix F:

$$\widehat{C}_r = FC_r^+F = FC_r^+F^{\mathrm{T}}$$

where $\widehat{C}$ is the new *repaired* correlation matrix—i.e., it is positive definite, symmetric, and has unit diagonal elements. To achieve this, the diagonal elements of F must be given by:

$$F_{ii} = \frac{1}{\sqrt{C_{r_{ii}}^+}}$$

We thus have:

$$\begin{aligned} \widehat{C}_r &= FC_r^+F^{\mathrm{T}} \\ &= FVK(VK)^{\mathrm{T}}F^{\mathrm{T}} \\ &= (FVK)\left(K^{\mathrm{T}}V^{\mathrm{T}}F^{\mathrm{T}}\right) \\ &= (FVK)(FVK)^{\mathrm{T}} \end{aligned} \tag{3.4.1}$$

An *optimally repaired* correlation matrix C_r^*, which minimizes the distance $\|C_r - C_r^*\|$, can be obtained via numerical optimization on the n-dimensional unit hypersphere; this is described below.

However, it has been found that $\widehat{C}_r$ is a very good approximation for the optimal estimate C_r^*.

Optimally repaired correlation matrix

Here we provide details of how to obtain an optimally repaired correlation matrix by using hyperspherical coordinates and the method of Rebonato and Jäckel (1999/2000)—for a different approach see Higham (2002) or Qi and Sun (2006).

The Cartesian coordinates of the ith point on an n-dimensional hypersphere with radius r can be shown to be:

$$x_{i,1} = r\cos(\theta_{i,1})$$
$$x_{i,2} = r\sin(\theta_{i,1})\cos(\theta_{i,2})$$

$$
\begin{aligned}
x_{i,3} &= r\sin(\theta_{i,1})\sin(\theta_{i,2})\cos(\theta_{i,3}) \\
x_{i,4} &= r\sin(\theta_{i,1})\sin(\theta_{i,2})\sin(\theta_{i,3})\cos(\theta_{i,4}) \\
&\vdots \\
x_{i,n-1} &= r\sin(\theta_{i,1})\sin(\theta_{i,2})\cdots\sin(\theta_{i,n-3})\sin(\theta_{i,n-2})\cos(\theta_{i,n-1}) \\
x_{i,n} &= r\sin(\theta_{i,1})\sin(\theta_{i,2})\cdots\sin(\theta_{i,n-2})\sin(\theta_{i,n-1})
\end{aligned}
$$

where $\theta_{i,1}$ are spherical coordinates and have the following constraints: $0 \leqslant \theta_{i,k} \leqslant \pi$, $k = 1, \ldots, n-2$, and $0 \leqslant \theta_{i,n-1} \leqslant 2\pi$.

By construction the radius of the sphere satisfies

$$r^2 = \sum_{k=1}^{n} (x_{i,k})^2$$

This can be seen as follows:

$$
\begin{aligned}
&\sum_{k=1}^{n} (x_{i,k})^2 \\
&\quad = r^2\big\{\cos^2(\theta_{i,1}) + \sin^2(\theta_{i,1})\big(\cos^2(\theta_{i,2}) \\
&\qquad\qquad + \sin^2(\theta_{i,2})\big(\cos^2(\theta_{i,3}) \\
&\qquad\qquad + \sin^2(\theta_{i,3})\big(\cos^4(\theta_{i,4}) \\
&\qquad\qquad + \cdots + \sin^2(\theta_{i,n-2}) \\
&\qquad\qquad\qquad \times \big(\cos^2(\theta_{i,n-1}) + \sin^2(\theta_{i,n-1})\big)\cdots\big)\big\} \\
&\quad = r^2
\end{aligned}
$$

where we have used

$$\cos^2(\theta_{i,k}) + \sin^2(\theta_{i,k}) = 1, \quad k = 1, \ldots, n-1$$

If, when $r = 1$, the coordinates of n hyperspherical points are stored in the n rows of the n by n matrix B^{T}, then:

$$
\begin{aligned}
B^{\mathrm{T}}_{i,1} &= \cos(\theta_{i,1}) \\
B^{\mathrm{T}}_{i,j} &= \cos(\theta_{i,j}) \prod_{k=1}^{j-1} \sin(\theta_{i,k}), \quad j = 2, \ldots, n-1, \\
B^{\mathrm{T}}_{i,n} &= \prod_{k=1}^{n-1} \sin(\theta_{i,k}), \quad n > 1, \\
B^{\mathrm{T}} &= \begin{pmatrix}
\cos(\theta_{1,1}) & \sin(\theta_{1,1})\cos(\theta_{1,2}) & \sin(\theta_{1,1})\sin(\theta_{1,2})\cos(\theta_{1,3}) & \cdots \\
\cos(\theta_{2,1}) & \sin(\theta_{2,1})\cos(\theta_{2,2}) & \sin(\theta_{2,1})\sin(\theta_{2,2})\cos(\theta_{2,3}) & \cdots \\
\cos(\theta_{3,1}) & \sin(\theta_{3,1})\cos(\theta_{3,2}) & \sin(\theta_{3,1})\sin(\theta_{3,2})\cos(\theta_{3,3}) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}
\end{aligned}
$$

It can thus be seen that the diagonal elements of $B^{\mathrm{T}}B$ are

$$\left(B^{\mathrm{T}}B\right)_{i,i} = \sum_{k=1}^{n} (x_{i,k})^2 = 1$$

The Cholesky decomposition can be formed by setting the angles of the upper triangular elements of B^{T} to zero, and this results in

$$L^{\mathrm{T}} = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ \cos(\theta_{2,1}) & \sin(\theta_{2,1}) & 0 & \cdots \\ \cos(\theta_{3,1}) & \sin(\theta_{3,1})\cos(\theta_{3,2}) & \sin(\theta_{3,1})\sin(\theta_{3,2}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$L^{\mathrm{T}}_{1,1} = 1$$

$$L^{\mathrm{T}}_{i,1} = \cos(\theta_{i,1})$$

$$L^{\mathrm{T}}_{i,j} = \cos(\theta_{i,j}) \prod_{k=1}^{j-1} \sin(\theta_{i,k}), \quad j = 2, \ldots, i-1,$$

$$L^{\mathrm{T}}_{i,i} = \prod_{k=1}^{i-1} \sin(\theta_{i,k}), \quad i > 1,$$

$$L^{\mathrm{T}}_{i,j} = 0, \quad j = i+1, \ldots, n$$

We want to find the positive definite matrix C_r^* which minimizes $\|C_r - C_r^*\|$. This can be found by writing

$$C_r^* = L^{\mathrm{T}}L$$

and using numerical optimization to determine the appropriate $n(n-1)/2$ angles. An initial approximation can be obtained by computing the Cholesky factorization $\widehat{C}_r = \widehat{L}^{\mathrm{T}}\widehat{L}$ and then calculating the angles corresponding to each nonzero element of $\widehat{L}^{\mathrm{T}}$.

Covariance matrix

We will now consider the case when a covariance matrix C is supplied which is not positive definite—that is, there is no Cholesky decomposition $C = L^{\mathrm{T}}L$, where L is lower triangular.

In these circumstances, since a covariance matrix does not require unit diagonal elements, it is possible to repair C using:

$$C^+ = VK(VK)^{\mathrm{T}}$$

where V and K have the same meanings as before. A better approximation could be obtained via numerical optimization of the elements of the Cholesky decomposition. However, these optimal points are no longer constrained to lie on the n-dimensional unit hypersphere.

3.4.3 Normal distribution

Here we show how to generate a p variate normal distribution with a given mean and covariance matrix.

We will denote the vector containing the variates of the ith observation from a p variate zero mean normal distribution by Z_i; that is, we write a sample of n observations as:

$$Z_i \sim \mathrm{N}(0, C), \quad i = 1, \ldots, n, \tag{3.4.2}$$

where C is the $p \times p$ covariance matrix.

Further $Z_{i,k}$ is used to denote the kth element of Z_i, which contains the value of the kth variate for the ith observation.

From a computational point of view, we can then consider a sample of n observations to be represented by the $n \times p$ matrix Z. The ith row of Z contains the values for ith the observation, and the kth column of the ith row, $Z_{i,k}$, contains the value of the kth variate for the ith observation.

Also, since the distribution has zero mean, the sample covariance matrix is given by: $C = Z^{\mathrm{T}}Z$.

To generate variates with the covariance matrix C we can use the fact that, if the matrix C is positive definite, a Cholesky factorization exists in which:

$$C = AA^{\mathrm{T}} \tag{3.4.3}$$

where A is lower triangular.

We can therefore generate p variates which have a covariance matrix C as follows.

First generate by (for example) using the Box–Muller method described in Section 3.3.1, the independent normal variates:

$$X \sim \mathrm{N}(0, I_p)$$

where the vector X contains the p variates, I_p is the unit matrix, and $XX^{\mathrm{T}} = I_p$.

Then, using the Cholesky factorization of Eq. (3.4.4), form:

$$Y = AX \tag{3.4.4}$$

where Y is a p-element vector.

Now, since $YY^{\mathrm{T}} = AX(AX)^{\mathrm{T}} = A(XX^{\mathrm{T}})A^{\mathrm{T}} = AA^{\mathrm{T}} = C$, we have that

$$Y \sim \mathrm{N}(0, C)$$

Variates that have nonzero means $\mu_k, k = 1, \ldots, p$, can be obtained by simply modifying Eq. (3.4.4) to:

$$Y' = AX + \mu \tag{3.4.5}$$

where Y' is a p variate vector that is distributed as $\mathrm{N}(\mu, C)$, and the p elements of vector μ contain the means of the variates $Y'_k, k = 1, \ldots, p$.

If the matrix C is not positive definite, then we can create a repaired matrix, $\widehat{C}$, by using the approach outlined in Section 3.4.2.

We now use the decomposition:

$$\widehat{C} = VK(VK)^{\mathrm{T}}$$

Under these circumstances the p-element vectors Y and Y' are generated using the following modified versions of Eqs. (3.4.4) and (3.4.5):

$$Y = VKX \quad \text{and} \quad Y' = VKX + \mu \tag{3.4.6}$$

The method for generating variates Y from a given correlation matrix C_r is identical. However, in this case nonpositive definite matrices are repaired as $\widehat{C}_r = FVK(FVK)^{\mathrm{T}}$; see Section 3.4.2.

A function to generate correlated normal and lognormal variates is given in Code excerpt 3.2.

```
  long Quasirandom_Normal_LogNormal_Correlated(long fcall, long seq, long lnorm,
            double means[], long n, double c[], long tdc, double tol, long *irank,
            double x[], double work[], long lwk) {

/* Input parameters:
   =================
   fcall    -  if fcall == 1 then it is an initialisation call,
               if fcall == 0 then a continuation call
   seq      -  if seq == 0 then a Faure sequence, if seq == 1 then a Niederreiter sequence,
               if seq == 2 then a Sobol sequence
   lnorm    -  if lnorm == 1 then it is a lognormal distribution,
               if lnorm == 0 then a normal distribution
   n        -  the number of variates, n must be less than 40
   c[]      -  a matrix which contains the required covariance matrix, C
   tdc      -  the second dimension of the matrix C
   tol      -  the tolerance used for calculating the rank of the covariance matrix C
   means[]  -  the means of the independent normal variates
   std[]    -  the standard deviations of the independent normal variates
   lwk      -  the size of the work array, work

   Output parameters:
   ==================
   rank     -  the computed rank of the covariance matrix C
   x[]      -  the elements x[0], .. x[n-1] contain the variates

   Input/Output parameters:
   ========================
   work     -   a work array
*/

  double zero = 0.0, one = 1.0, two = 2.0;
  long n1, i, j, k, kk;
  double mtol, alpha;
  long ptrc, ptre, ptrv, ptrw, ptrw0, ptrw1;

#define C(I,J) c[((I)-1) * tdc + ((J)-1)]
#define MEANS(I) means[(I)-1]
#define X(I) x[(I)-1]
#define WORK(I) work[(I)-1]

      if (lwk < (2 + 3*n + 2*n*n + 3)) {
         printf ("Error lwk is too small \n");
         return 1;
      }

      ptre = 2;
      ptrv = n+2;
      ptrw = n*n + n + 2;
```

Code excerpt 3.2 The function `Quasirandom_Normal_LogNormal_Correlated` which generates correlated quasi-random normal variates and correlated quasi-random lognormal variates.

```
/* add extra 1 to allow for odd values of n */
      ptrw0 = ptrw + 1 + n;
      ptrw1 = ptrw0 + 1 + n;
      ptrc  = ptrw1 + n + 1;
      n1 = n;
      if (((n/2)*2) != n) { /*  test for odd n */
          n1 = n + 1;
      }
      if (fcall) {  /* first call for initialisation */
         if (lnorm) { /* lognormal distribution */
            for (i = 1; i <=n; ++i) { /* Load the modified covariance matrix into WORK */
               for (j = 1; j <= n; ++j) {
                 WORK(ptrc+(i-1)*n+j-1) = log(one + C(i,j)/(MEANS(i)*MEANS(j)));
               }
            }
         }
         else { /* normal distribution */
            for (i = 1; i <=n; ++i) {  /* Load the covariance matrix into WORK */
                for (j = 1; j <= n; ++j) {
                   WORK(ptrc+(i-1)*n+j-1) = C(i,j);
                }
            }
         }

         /* calculate the eigenvalues and eigenvector of the matrix that
            has been loaded into WORK */
         calc_eigvals_eigvecs (n,&WORK(ptrc),n,&WORK(ptre),&WORK(ptrv),n);
         /* The code uses NAG routine f02abc */
         *irank = 0;
/*       printf ("The eigenvalues are \n");
         for (j=n; j >= 1; --j) {
            printf ("%12.5f \n", WORK(ptre+j-1));
         }
*/
         for (j=n; j >= 1; --j) { /* use the eigenvalues to calculate the rank of the matrix */
           if (WORK(ptre+j-1) < tol) goto L24;
           *irank = *irank + 1;
         }
         printf ("*irank = %ld \n",*irank);
       L24:
         mtol = -tol;
         if (WORK(ptre) < mtol) {
            printf ("Warning there is an eigenvalue less than %12.4f \n",mtol);
         }
         for (j=1; j <= *irank; ++j) {
            kk = 1;
            for (k=1; k <=n; ++k) {
              if(WORK(ptrv+(k-1)*n+(j-1)) != zero) goto L28;
              kk = kk + 1;
            }
          L28:
          /* ensure that all eigenvectors have the same sign on different machines */
           alpha = sqrt(WORK(ptre+j-1));
           if  (WORK(ptrv+(kk-1)*n+(j-1)) < zero)  alpha = -sqrt(WORK(ptre+j-1));
           for (i = 1; i <= n; ++i) {
             WORK(ptrv+(j-1)+(i-1)*n)=WORK(ptrv+(j-1)+(i-1)*n)*alpha;
           }
          }
/*        printf ("The eigenvectors are \n");
          for (j=1; j <= *irank; ++j) {
             for (i = 1; i <= n; ++i) {
               printf ("%10.5f ", WORK(ptrv+(j-1)+(i-1)*n));
             }
             printf ("\n");
          }
*/
         for (i = 1; i <=n; ++i)
         { /* store a vector of ones and zeros for generating the quasi-random numbers */
            WORK(ptrw0+i-1) = zero;
            WORK(ptrw1+i-1) = one;
         }
```

Code excerpt 3.2 (*Continued*).

```
         for (i = n; i <= n1; ++ i) {
            WORK(ptrw0+i-1) = zero;
            WORK(ptrw1+i-1) = one;
         }
      } /* end of first  call section */

  /* generate a vector of n1 random variables from a standard normal distribution,
     zero mean and unit variance */
      Quasi_Normal_Independent(fcall, seq, &WORK(ptrw0), &WORK(ptrw1),  n1, &WORK(ptrw));

/*    printf ("The quasi random numbers are:\n");
      for (i = 1; i <= n; ++i) {
          printf ("%12.4f \n", WORK(ptrw+(i-1)));
      }
*/
   /* Now generate variates with the specified mean and variance */
      if (lnorm) { /* a lognormal distribution */
         for (i = 1; i <= n; ++i) {
           X(i) =  log(MEANS(i)) - WORK(ptrc+(i-1)*n+i-1)/two;
           for (k = 1; k <= *irank; ++k) {
               X(i)=X(i)+WORK(ptrv+(k-1)+(i-1)*n)*WORK(ptrw+k-1);
           }
         }
         for (i = 1; i <= n; ++i) {
             X(i) = exp(X(i));
         }
      }
     else { /* a normal distribution */
          for (i = 1; i <= n; ++i) {
            X(i) =  MEANS(i);
            for (k = 1; k <= *irank; ++k) {
              X(i)=X(i)+WORK(ptrv+(k-1)+(i-1)*n)*WORK(ptrw+k-1);
            }
          }
      }
/*    printf ("The generated variates are:\n");
      for (i = 1; i <= n; ++i) {
          printf (" %12.4f \n", X(i));
      }
*/
      return 0;
    }
```

Code excerpt 3.2 (*Continued*).

In order to visualize the effect of the covariance matrix we will display the results of using function `Quasirandom_Normal_LogNormal_Correlated` to generate the following variates:

- A vector of three normal independent variates with covariance matrix:

$$C_1 = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

- A vector of three normal variates in which the elements of the covariance matrix are all positive; the covariance matrix is:

$$C_2 = \begin{pmatrix} 1.0 & 0.8 & 0.8 \\ 0.8 & 1.0 & 0.8 \\ 0.8 & 0.8 & 1.0 \end{pmatrix}$$

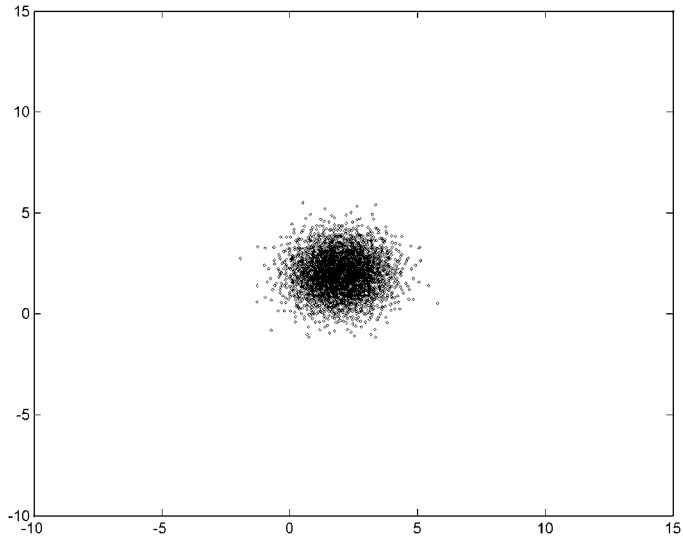

Figure 3.5 Scatter diagram for a sample of 3000 observations ($Z_i, i = 1, \ldots, 3000$) generated from a multivariate normal distribution consisting of three variates with covariance matrix C_1 and mean μ. Here we plot the values of the first variate against the values of the second variate. If we use the notation of Eq. (3.4.2), then the (x, y) coordinates for the points are $x_i = Z_{i,1}$, $i = 1, \ldots, 3000$, and $y_i = Z_{i,2}$, $i = 1, \ldots, 3000$.

- A vector of three normal variates in which two elements of the covariance matrix are negative; the covariance matrix is:

$$C_3 = \begin{pmatrix} 1.0 & -0.7 & 0.2 \\ -0.7 & 1.0 & 0.2 \\ 0.2 & 0.2 & 1.0 \end{pmatrix}$$

In all cases the mean vector is given by:

$$\mu = \begin{pmatrix} 2.0 \\ 2.0 \\ 2.0 \end{pmatrix}$$

The results are displayed in Figs. 3.5–3.7.

3.4.4 Lognormal distribution

The multivariate lognormal distribution is important because it is the asset returns distribution assumed by the Black–Scholes equation.

Let the p-element vectors Y and X be related by $Y = \log(X)$ where $Y \sim \mathrm{N}(\mu, \Sigma)$, μ is a p-element vector, and Σ is a $p \times p$ matrix. Then $X = \exp(Y)$ and X has multivariate lognormal distribution, which we denote by $X \sim \Lambda(\mu, \Sigma)$.

We will represent the p-element mean vector of X by $\bar{m}$ and the $p \times p$ covariance matrix of X by $\mathcal{S}$.

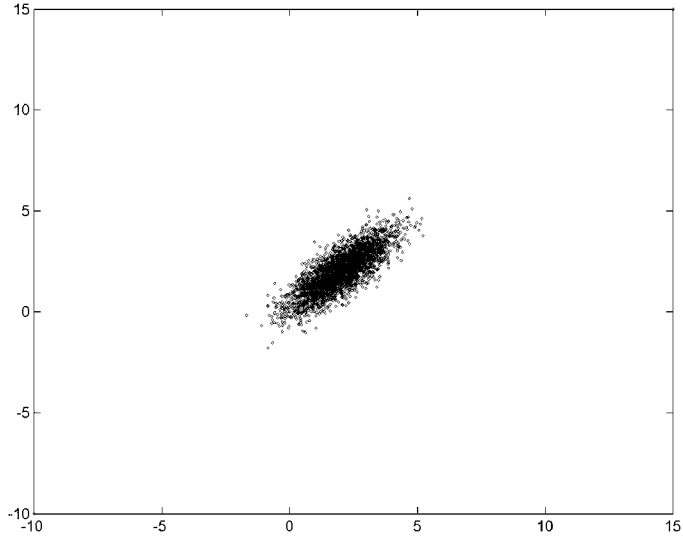

Figure 3.6 Scatter diagram for a sample of 3000 observations ($Z_i, i = 1, \ldots, 3000$) generated from a multivariate normal distribution consisting of three variates with covariance matrix C_2 and mean μ. Here we plot the values of the first variate against the values of the second variate. If we use the notation of Eq. (3.4.2), then the (x, y) coordinates for the points are $x_i = Z_{i,1}$, $i = 1, \ldots, 3000$, and $y_i = Z_{i,2}$, $i = 1, \ldots, 3000$.

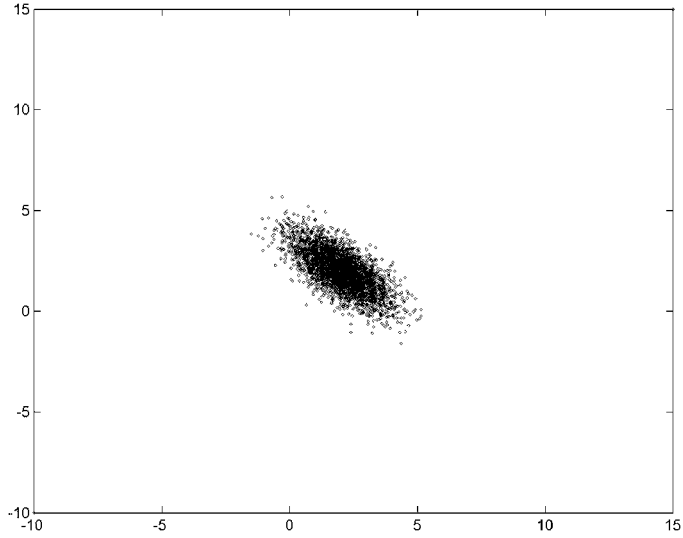

Figure 3.7 Scatter diagram for a sample of 3000 observations ($Z_i, i = 1, \ldots, 3000$) generated from a multivariate normal distribution consisting of three variates with covariance matrix C_3 and mean μ. Here we plot the values of the first variate against the values of the second variate. If we use the notation of Eq. (3.4.2), then the (x, y) coordinates for the points are $x_i = Z_{i,1}$, $i = 1, \ldots, 3000$, and $y_i = Z_{i,2}$, $i = 1, \ldots, 3000$.

It can be shown that:

$$\Sigma_{i,j} = \log\left(1 + \frac{\mathcal{S}_{i,j}}{\bar{m_i}\bar{m_j}}\right) \tag{3.4.7}$$

and

$$\mu_i = \log(\bar{m_i}) - \frac{\Sigma_{i,i}}{2}, \quad i = 1, \ldots, p, \text{ and } j = 1, \ldots, p \tag{3.4.8}$$

For the case of independent variates we then have:

$$\mu_i = \log(\bar{m_i}) - \frac{\sigma_i^2}{2}, \quad i = 1, \ldots, p,$$

and

$$\Sigma_{i,i} = \sigma_i^2 = \log\left(1 + \frac{s_i^2}{\bar{m_i}^2}\right), \quad i = 1, \ldots, p, \quad \text{and for } i \neq j, \quad \Sigma_{i,j} = 0$$

which are just Eqs. (3.3.13) and (3.3.14) given in Section 3.3.2.

Code excerpt 3.3 shows how to generate a multivariate lognormal distribution with a given mean $\bar{m}$ and covariance matrix $\mathcal{S}$. More complete information can be found in the function `Quasirandom_Normal_LogNormal_Correlated` which is provided in Code excerpt 3.2.

```
double sig[40][40], s[40][40]; /* limit of 40 */
double means[40], x[40], lx[40], tmp;
        .   .   .
#define S(I,J) s[(I)-1][(J)-1]
#define SIG(I,J) sig[(I)-1][(J)-1]
#define MEANS(i) means[(I)-1]    /* the means of the lognormal distribution */
#define X(I) x[(I)-1] /* normal variates */
#define LX(I) lx[(I)-1] /* lognormal variates */

      .   .   .
/* obtain the Gaussian covariance matrix SIG, that corresponds to the
   lognormal covariance matrix S. */

  for (i=1; i <= m; ++i) {
      for (j=1; j <= m; ++j) {
        tmp = MEANS(i) * MEANS(j);
        SIG(i,j) = log( 1 + (S(i,j)/tmp));
      }
  }
      .   .   .
/* Generate multivariate Gaussian variates X(i),i = 1,..,m, with zero mean and
   covariance matrix SIG, using section ..   */
      .   .   .
/* Using equation () generate normal variates with the correct mean */
  for (i=1; i <= m; ++i) {
     X(i) = X(i) + log(MEANS(i)) - SIG(i,i)/2;
  }
/* Now exponentiate to create lognormal variates with mean
   XMEAN, and covariance matrix S */
  for (i=1; i <= m; ++i) {
      LX(i) = exp(X(i));
  }
```

Code excerpt 3.3 Illustrating how to generate variates from a lognormal distribution with a given mean and covariance matrix.

4 European options

4.1 Introduction

A European option taken out at current time t gives the owner the right (but not the obligation) to do *something* when the option *matures* at time T. This could, for example, be the right to buy or sell stocks at a particular *strike* price. The option would of course only be exercised if it was in the owner's interest to do so. For example, a single asset European *vanilla put* option, with strike price E and *expiry* time T, gives the owner the right at time T to sell a particular asset for E. If the asset is worth S_T at maturity then the value of the put option at maturity, known as the *payoff*, is thus $\max(E - S_T, 0)$. By contrast a single asset European vanilla *call* option, with strike price E and expiry time T, gives the owner the right at time T to buy an asset for E; the payoff at maturity for a call option is $\max(S_T - E, 0)$.

The owner of an American option has the right (but not the obligation) to exercise the option *at any time* from current time t to option maturity. These options are more difficult to value than European options because of this extra flexibility. Even the *simple* single asset American vanilla put has no analytic solution and requires finite-difference and lattice methods to estimate its value. Many European options, on the other hand, take the form of a *relatively easy* definite integral from which it is possible to compute a closed form solution. The valuation of multiasset European options, dependent on a large number of underlying assets, is more complicated but can conveniently be achieved by using Monte Carlo simulation to compute the required multidimensional definite integral.

The expected *current value* of a single asset European vanilla option will depend on the current asset price at time t, S, the duration of the option, $\tau = T - t$, the strike price, E, the riskless interest rate, r, and the probability density function of the underlying asset at maturity, $p(S_T)$.

4.2 Pricing derivatives using a martingale measure

In this section we will briefly summarize the results of Harrison and Kreps (1979) and Harrison and Pliska (1981). Let us consider an economy over the time interval $[0, T]$ which consists of n assets $S^i, i = 1, \ldots, n$, which can take

the values $S_t^i, i = 1, \ldots, n, 0 \leqslant t \leqslant T$. Any asset S^i which only takes values that are greater than zero is called a *numeraire*. Numeraires can be used to denominate all the asset prices in the economy. So (for example) if S^1 is a numeraire then the prices of the other assets denominated in units of S_1 are the relative prices $Z_t^i = (S_t^i / S_t^1), i = 2, \ldots, n$.

One can find a unique probability measure $\mathbb{Q}$ such that the relative prices $Z_t^i, i = 2, \ldots, n$, are martingales. If the economy is free of arbitrage opportunities then every payoff pattern H_T can be represented as a linear combination of the asset values $S_t^i, i = 1, \ldots, n$, and in addition the relative price process (H_T / S_T^1) is a martingale.

This means that we can write

$$E^{\mathbb{Q}}\left[\frac{H_t}{S_t^1}\right] = E^{\mathbb{Q}}\left[\frac{H_T}{S_T^1}\right], \quad \text{where } 0 \leqslant t \leqslant T$$

The current (time t) value V_t of the payoff H_T is thus

$$V_t = S_t^1 E^{\mathbb{Q}}\left[\frac{H_T}{S_T^1}\right]$$

In general for a numeraire N which takes the values $N_t, 0 \leqslant t \leqslant T$, we can write

$$V_t = N_t E^{\mathbb{Q}}\left[\frac{H_T}{N_T}\right] \tag{4.2.1}$$

Equation (4.2.1) is very important because V_t is the current (time t) price of a financial derivative with maturity T and payoff H_T.

It should be mentioned that the price of a financial derivative is independent of the martingale measure under which it is valued, and thus the same price V_t will be obtained for different numeraires N.

4.3 Put call parity

4.3.1 Discrete dividends

Here we consider single asset European put and call options, and derive the following relationship between their values in the presence of cash dividends:

$$c(S, E, \tau) + E\exp(-r\tau) + D = p(S, E, \tau) + S \tag{4.3.1}$$

where D is the *present value* of the dividends that are paid during the life of the option. That is:

$$D = \sum_{k=1}^{n} D_k \exp\big(-r(t_k - t)\big)$$

with D_k the kth cash dividend paid at time t_k; the other symbols have already been defined in Section 4.1.

This result can be proved by considering the following two investments:

PORTFOLIO A: One European call, $c(S, E, \tau)$, and cash of value $E\exp(-r\tau)+D$.

PORTFOLIO B: One European put, $p(S, E, \tau)$, and one share of value S.

At option maturity the value of the call and put are $c(S_T, E, 0)$ and $p(S_T, E, 0)$, respectively; also at time T the value of the dividends paid during the life of the option is $D\exp(r\tau)$.

We now consider the value of both portfolios at time T under all possible conditions.

If $S_T \geqslant E$

Portfolio A is worth:

$$\begin{aligned}\max(S_T - E, 0) + \exp(r\tau)\{E\exp(-t\tau) + D\} &= S_T - E + E + D\exp(r\tau)\\ &= S_T + D\exp(r\tau)\end{aligned}$$

Portfolio B is worth:

$$\begin{aligned}\max(E - S_T, 0) + S_T + D\exp(r\tau) &= 0 + S_T + D\exp(r\tau)\\ &= S_T + D\exp(r\tau)\end{aligned}$$

If $S_T < E$

Portfolio A is worth:

$$\begin{aligned}\max(S_T - E, 0) + \exp(r\tau)\{E\exp(-t\tau) + D\} &= 0 + E + D\exp(r\tau)\\ &= E + D\exp(r\tau)\end{aligned}$$

Portfolio B is worth:

$$\begin{aligned}\max(E - S_T, 0) + S_T + D\exp(r\tau) &= E - S_T + S_T + D\exp(r\tau)\\ &= E + D\exp(r\tau)\end{aligned}$$

We have therefore shown that under all conditions the value of portfolio A is the same as that of portfolio B.

4.3.2 Continuous dividends

Here we consider single asset European put and call options, and derive the following relationship:

$$c(S, E, \tau) + E\exp(-r\tau) = p(S, E, \tau) + S\exp(-q\tau) \tag{4.3.2}$$

where q is the asset's continuous dividend yield that is paid during the life of the option. The result can be proved by considering the following two investments:

PORTFOLIO A: One European call, $c(S, E, \tau)$, and cash of value $E\exp(-r\tau)$.

PORTFOLIO B: One European put, $p(S, E, \tau)$, and one share of value $S\exp(-q\tau)$.

At option expiry the value of the call and put are $c(S_T, E, 0)$ and $p(S_T, E, 0)$, respectively. Also, if the value of the share at time t is denoted by S, the combined value of shares and dividends at time T is $S\exp(q\tau)$: Note that q is treated in a similar manner to the continuously compounded riskless interest rate r.

If $S_T \geqslant E$

Portfolio A is worth:

$$\max(S_T - E, 0) + \exp(r\tau)E\exp(-r\tau) = S_T - E + E = S_T$$

Portfolio B is worth:

$$\max(E - S_T, 0) + S_T\exp(-q\tau)\exp(q\tau) = 0 + S_T = S_T$$

where $S_T\exp(-q\tau)\exp(q\tau)$ is the combined value of the shares and dividends at option maturity.

If $S_T < E$

Portfolio A is worth:

$$\max(S_T - E, 0) + \exp(r\tau)E\exp(-r\tau) = 0 + E = E$$

Portfolio B is worth:

$$\max(E - S_T, 0) + S_T\exp(-q\tau)\exp(q\tau) = E - S_T + S_T = E$$

We have therefore shown that under all conditions the value of portfolio A is the same as that of portfolio B.

4.4 Vanilla options and the Black–Scholes model

4.4.1 *The option pricing partial differential equation*

In this section we will derive the (Black–Scholes) partial differential equation that is obeyed by options written on a single asset.

Previously, in Section 2.3 and Section 2.5, we derived Ito's lemma, which provides an expression for the change in value of the function $\phi(X, t)$, where X is a stochastic variable. When the stochastic variable, X, follows GBM, the change in the value of ϕ was shown to be given by Eq. (2.3.6). Here we will assume that the function $\phi(S, t)$ is the value of a financial option and that the price of the underlying asset, S, follows GBM.

If we denote the value of the financial derivative by f, then its change, $\mathrm{d}f$, over the time interval $\mathrm{d}t$ is given by:

$$\mathrm{d}f = \left(\mu S\frac{\partial f}{\partial S} + \frac{\partial f}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2}\right)\mathrm{d}t + \frac{\partial f}{\partial S}\sigma S\,\mathrm{d}W, \quad \mathrm{d}W \sim \mathrm{N}(0, \mathrm{d}t)$$

The discretized version of this equation is:

$$\Delta f = \Delta t\left(\mu S\frac{\partial f}{\partial S} + \frac{\partial f}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2}\right) + \frac{\partial f}{\partial S}\sigma S\Delta W, \quad \Delta W \sim \mathrm{N}(0, \Delta t), \tag{4.4.1}$$

where the time interval is now Δt and the change in derivative value is Δf.

If we assume that the asset price, S, follows GBM we also have:

$$\Delta S = \mu S\Delta t + \sigma S\Delta W, \quad \Delta W \sim \mathrm{N}(0, \Delta t), \tag{4.4.2}$$

where μ is the constant drift and the definition of the other symbols is as before. Let us now consider a portfolio consisting of -1 derivative and $\frac{\partial f}{\partial S}$ units of the underlying stock. In other words we have gone *short* (that is sold) a derivative on an asset and have $\frac{\partial f}{\partial S}$ stocks of the (same) underlying asset. The value of the portfolio, Π, is therefore:

$$\Pi = -f + \frac{\partial f}{\partial S}S \tag{4.4.3}$$

and the change, $\Delta\Pi$, in the value of the portfolio over time Δt is:

$$\Delta\Pi = -\Delta f + \frac{\partial f}{\partial S}\Delta S \tag{4.4.4}$$

Substituting Eqs. (4.4.1) and (4.4.2) into Eq. (4.4.4) we obtain:

$$\begin{aligned}\Delta\Pi &= -\left(\mu S\frac{\partial f}{\partial S} + \frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2}\right)\Delta t \\ &\quad - \sigma S\Delta W\frac{\partial f}{\partial S} + \frac{\partial f}{\partial S}\{\mu S\Delta t + \sigma S\Delta W\} \\ \therefore \Delta\Pi &= -\mu S\Delta t\frac{\partial f}{\partial S} - \Delta t\frac{\partial f}{\partial t} - \frac{1}{2}\Delta t\sigma^2 S^2\frac{\partial^2 f}{\partial S^2} \\ &\quad - \sigma S\Delta W\frac{\partial f}{\partial S} + \mu S\Delta t\frac{\partial f}{\partial S} + \sigma S\Delta W\frac{\partial f}{\partial S}\end{aligned} \tag{4.4.5}$$

Cancelling terms we obtain:

$$\Delta\Pi = -\Delta t\left\{\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2}\right\} \tag{4.4.6}$$

If this portfolio is to grow at the riskless interest rate r, we have:

$$r\Pi\Delta t = \Delta\Pi$$

So we have that:

$$r\Pi\Delta t = -\Delta t\left\{\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2}\right\} \tag{4.4.7}$$

Substituting for Π, we obtain:

$$r\Delta t\left(f - S\frac{\partial f}{\partial S}\right) = -\Delta t\left\{\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2}\right\} \tag{4.4.8}$$

On rearranging we have:

$$\frac{\partial f}{\partial t} + S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \tag{4.4.9}$$

which is the Black–Scholes partial differential equation. See Appendix I concerning the Feynman–Kac formula.

Let us now consider put and call options on the same underlying asset. If we let c be the value of a European call option and p that of a European put option then we have the following equations:

$$\frac{\partial p}{\partial t} + S\frac{\partial p}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 p}{\partial S^2} = rp \tag{4.4.10}$$

and

$$\frac{\partial c}{\partial t} + S\frac{\partial c}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 c}{\partial S^2} = rc \tag{4.4.11}$$

If we now form a linear combination of put and call options, $\Psi = a_1 c + a_2 p$, where both a_1 and a_2 are constants, then Ψ also obeys the Black–Scholes equation:

$$\frac{\partial \Psi}{\partial t} + S\frac{\partial \Psi}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \Psi}{\partial S^2} = r\Psi \tag{4.4.12}$$

We will now prove that Ψ satisfies Eq. (4.4.12).

First we rewrite Eq. (4.4.12) as:

$$\begin{aligned}&\frac{\partial (a_1 c + a_2 p)}{\partial t} + S\frac{\partial (a_1 c + a_2 p)}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 (a_1 c + a_2 p)}{\partial S^2} \\ &\quad = r(a_1 c + a_2 p)\end{aligned} \tag{4.4.13}$$

and use the following results from elementary calculus:

$$\frac{\partial (a_1 c + a_2 p)}{\partial t} = a_1 \frac{\partial c}{\partial t} + a_2 \frac{\partial p}{\partial t}$$

$$\frac{\partial (a_1 c + a_2 p)}{\partial S} = a_1 \frac{\partial c}{\partial S} + a_2 \frac{\partial p}{\partial S}$$

and

$$\frac{\partial^2 (a_1 c + a_2 p)}{\partial S^2} = a_1 \frac{\partial^2 c}{\partial S^2} + a_2 \frac{\partial^2 p}{\partial S^2}$$

If we denote the left-hand side of Eq. (4.4.12) by *LHS*, then we have:

$$\begin{aligned}LHS = a_1 &\left\{ \frac{\partial c}{\partial t} + S\frac{\partial c}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 c}{\partial S^2} \right\} \\ &+ a_2 \left\{ \frac{\partial p}{\partial t} + S\frac{\partial p}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 p}{\partial S^2} \right\}\end{aligned} \tag{4.4.14}$$

We now use Eqs. (4.4.10) and (4.4.11) to substitute for the values in the curly brackets in Eq. (4.4.14), and we obtain:

$$LHS = a_1 rc + a_2 rp \tag{4.4.15}$$

which is just the right-hand side of Eq. (4.4.12); so we have proved the result. It should be noted that this result is also true for American options, since they also obey the Black–Scholes equation.

The above result can be generalized to include a portfolio consisting of n single asset options. Here we have:

$$\Psi = \sum_{j=1}^{n} a_j f_j, \quad j = 1, \ldots, n,$$

where f_j represents the value of the jth derivative and a_j is the number of units of the jth derivative. To prove that Ψ follows the Black–Scholes equation we simply partition the portfolio into sectors whose options depend on the same underlying asset. We then proceed as before by showing that the value of each individual sector obeys the Black–Scholes equation and thus the value of the complete portfolio (the sum of the values of all the sectors) obeys the Black–Scholes equation. It should be mentioned that this result applies for both American and European options and it does not matter whether we have bought or sold the options.

In Chapter 5 we will use the fact that the difference between the value of a European option and the equivalent American option obeys the Black–Scholes equation. We can see this immediately by considering the following portfolios that are long in an American option and short (i.e., have sold) a European option:

$$\Psi^p = P - p, \qquad \Psi^c = C - c$$

where P and C are the values of American put and call options. Ψ^p and Ψ^c both obey the Black–Scholes equations, and are the respective differences in value of American/European put options and American/European call options.

4.4.2 The multiasset option pricing partial differential equation

In this section we will derive the multiasset (Black–Scholes) differential equation that is obeyed by options written on n assets. Proceeding as in Section 4.4.1, we will use the n-dimensional version of Ito's lemma to find the process followed by the value of a multiasset financial derivative. We will denote the value of this derivative by $f(\mathcal{S}, t)$, where $\mathcal{S}$ is an n-element stochastic vector containing the prices of the underlying assets, $S_i, i = 1, \ldots, n$. If we assume that $\mathcal{S}$ follows n-dimensional GBM then the change in the value of the derivative, $\mathrm{d}f$, is (see

Eq. (2.5.8)) given by:

$$\mathrm{d}f = \left\{ \frac{\partial f}{\partial t} + \sum_{i=1}^{n} \mu_i S_i \frac{\partial f}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_i \sigma_j S_i S_j \rho_{ij} \frac{\partial^2 f}{\partial S_i \, \partial S_j} \right\} \mathrm{d}t + \sum_{i=1}^{n} \frac{\partial f}{\partial S_i} \sigma_i S_i \, \mathrm{d}W_i \tag{4.4.16}$$

The discretized version of this equation is:

$$\Delta f = \left\{ \frac{\partial f}{\partial t} + \sum_{i=1}^{n} \mu_i S_i \frac{\partial f}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_i \sigma_j S_i S_j \rho_{ij} \frac{\partial^2 f}{\partial S_i \, \partial S_j} \right\} \Delta t + \sum_{i=1}^{n} \frac{\partial f}{\partial S_i} \sigma_i S_i \Delta W_i \tag{4.4.17}$$

where the time interval is now Δt and the change in derivative value is Δf.

Let us now consider a portfolio consisting of -1 derivative and $\frac{\partial f}{\partial S_i}$ units of the ith underlying stock. In other words we have gone *short* (that is sold) a derivative that depends on the price, $S_i, i = 1, \ldots, n$, of n underlying assets, and have $\frac{\partial f}{\partial S_i}$ units of the ith asset. The value of the portfolio, Π, is therefore:

$$\Pi = -f + \sum_{i=1}^{n} \frac{\partial f}{\partial S_i} S_i \tag{4.4.18}$$

and the change, $\Delta\Pi$, in the value of the portfolio over the time interval Δt is:

$$\Delta\Pi = -\Delta f + \sum_{i=1}^{n} \frac{\partial f}{\partial S_i} \Delta S_i \tag{4.4.19}$$

Since the stochastic variables $S_i, i = 1, \ldots, n$, follow n-dimensional GBM, the change in the ith asset price, ΔS_i, over the time interval Δt is given by:

$$\Delta S_i = \mu_i S_i \Delta t + \sigma_i S_i \Delta W_i, \quad i = 1, \ldots, n, \tag{4.4.20}$$

where $\Delta W_i = \mathrm{d}Z_i \sqrt{\Delta t}$,

$$E\left[\mathrm{d}Z_i^2\right] = 1, \quad i = 1, \ldots, n,$$

and

$$E[\mathrm{d}Z_i \, \mathrm{d}Z_j] = \rho_{i,j}, \quad i = 1, \ldots, n, \; j = 1, \ldots, n, \; i \neq j$$

Substituting Eqs. (4.4.17) and (4.4.20) into Eq. (4.4.19), we obtain:

$$\Delta\Pi = -\left\{ \frac{\partial f}{\partial t} + \sum_{i=1}^{n} \mu_i S_i \frac{\partial f}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i \, \partial S_j} \right\} \Delta t - \sum_{i=1}^{n} \sigma_i S_i \Delta W_i \frac{\partial f}{\partial S_i} + \sum_{i=1}^{n} \frac{\partial f}{\partial S_i} \{\mu_i S_i \Delta t + \sigma S_i \Delta W_i\}$$

$$\therefore \Delta\Pi = -\sum_{i=1}^{n} \mu_i S_i \Delta t \frac{\partial f}{\partial S_i} - \Delta t \frac{\partial f}{\partial t} - \frac{1}{2}\Delta t \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i\, \partial S_j}$$
$$- \sum_{i=1}^{n} \sigma_i S_i \Delta W_i \frac{\partial f}{\partial S_i} + \sum_{i=1}^{n} \mu_i S_i \Delta t \frac{\partial f}{\partial S_i}$$
$$+ \sum_{i=1}^{n} \sigma_i S_i \Delta W_i \frac{\partial f}{\partial S_i} \tag{4.4.21}$$

Cancelling terms we obtain:

$$\Delta\Pi = -\Delta t \left\{ \frac{\partial f}{\partial t} + \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i\, \partial S_j} \right\} \tag{4.4.22}$$

If this portfolio is to grow at the riskless interest rate r we have:

$$r\Pi\,\Delta t = \Delta\Pi$$

So from Eq. (4.4.22) we have that:

$$r\Pi\,\Delta t = -\Delta t \left\{ \frac{\partial f}{\partial t} + \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i\, \partial S_j} \right\} \tag{4.4.23}$$

Substituting for Π we obtain:

$$r\Delta t \left\{ f - \sum_{i=1}^{n} S_i \frac{\partial f}{\partial S_i} \right\}$$
$$= -\Delta t \left\{ \frac{\partial f}{\partial t} + \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i\, \partial S_j} \right\} \tag{4.4.24}$$

Rearranging Eq. (4.4.24) gives:

$$\frac{\partial f}{\partial t} + \sum_{i=1}^{n} S_i \frac{\partial f}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 f}{\partial S_i\, \partial S_j} = rf \tag{4.4.25}$$

which is the n-dimensional Black–Scholes partial differential equation.

4.4.3 The Black–Scholes formula

The Black–Scholes model consists of two assets: the riskless money account and an equity. It can be cast as the following two-dimensional Ito equation:

$$\begin{aligned} \mathrm{d}S_t &= \mu S_t\, \mathrm{d}t + \sigma S_t\, \mathrm{d}W^P \\ \mathrm{d}B_t &= r B_t\, \mathrm{d}t \end{aligned} \tag{4.4.26}$$

where W^P is Brownian motion (without drift) under measure $\mathbb{P}$, so $\mathrm{d}W^P \sim \mathrm{N}(0, \mathrm{d}t)$.

Current time will be denoted by t_0, and the option maturity time by T. The money market account has value $B_{t_0} = 1$ at time t_0 and $B_T = \exp(r(T - t_0)$ at time T.

We will now consider the process followed by the relative value $\phi(S_t, B_t) = S_t/B_t$.

Using the Ito quotient rule as described in Section 2.6.2 and substituting $X_1 = S_t$ and $X_2 = B_t$ in Eq. (2.6.8) we have:

$$\mathrm{d}\left(\frac{S_t}{B_t}\right) = \left(\frac{S_t}{B_t}\right)(\mu - r)\,\mathrm{d}t + \left(\frac{S_t}{B_t}\right)\sigma\,\mathrm{d}W^P$$

So finally we can write:

$$\mathrm{d}S_t^* = S_t^*(\mu - r)\,\mathrm{d}t + S_t^*\sigma\,\mathrm{d}W^P \tag{4.4.27}$$

where $S_t^* = S_t/B_t$.

Referring to Girsanov's theorem in Section 2.4, we can choose a probability measure $\mathbb{Q}$ such that:

$$\mathrm{d}W^P = \mathrm{d}W^Q - \left(\frac{\mu - r}{\sigma}\right)\mathrm{d}t \tag{4.4.28}$$

In Eq. (2.4.3) we thus have $k(t) = -((\mu - r)/\sigma)$ and

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} = \exp\left\{-\left(\frac{\mu - r}{\sigma}\right)W^P - \frac{1}{2}\left(\frac{\mu - r}{\sigma}\right)^2 t\right\} \tag{4.4.29}$$

See p. 114 of Musiela (1998). Substituting for $\mathrm{d}W^P$ in Eq. (4.4.27) yields

$$\mathrm{d}S_t^* = S_t^*\{\mu - r\}\,\mathrm{d}t - S_t^*\sigma\left(\frac{\mu - r}{\sigma}\right)\mathrm{d}t + S^*\sigma\,\mathrm{d}W^Q$$

which simplifies to

$$\mathrm{d}S_t^* = S_t^*\sigma\,\mathrm{d}W^Q \tag{4.4.30}$$

Equation (4.4.30) means that the process for S_t^* is a martingale under probability measure $\mathbb{Q}$.

Replacing $\mathrm{d}W^P$ in Eq. (4.4.26) with the value in Eq. (4.4.28) yields

$$\begin{aligned}\mathrm{d}S_t &= \mu S_t\,\mathrm{d}t + S_t\sigma\,\mathrm{d}W^P \\ &= \mu S_t\,\mathrm{d}t + S_t\sigma\left\{\mathrm{d}W^Q - \left(\frac{\mu - r}{\sigma}\right)\mathrm{d}t\right\} \\ &= \left\{S_t\mu\,\mathrm{d}t - S_t\sigma\left(\frac{\mu - r}{\sigma}\right)\right\}\mathrm{d}t + S_t\sigma\,\mathrm{d}W^Q\end{aligned}$$

So in the risk neutral measure $\mathbb{Q}$ the dynamics of $\mathrm{d}S$ are

$$\mathrm{d}S_t = S_t r\,\mathrm{d}t + S_t\sigma\,\mathrm{d}W^Q \tag{4.4.31}$$

Comparing Eq. (4.4.31) with the original Eq. (4.4.26) we see that changing from the real-world measure to the risk neutral measure simply involves substituting $\mathrm{d}W^Q$ for $\mathrm{d}W^P$ and r for μ.

We can now solve Eq. (4.4.31) by using the result given in Eq. (2.3.11). We have

$$S_T = S\exp\big(\nu(T-t_0) + \sigma W^{Q}_{T-t_0}\big)$$

where S is the asset price at current time t_0, and $\nu = r - \sigma^2/2$.

The forward price with maturity T, denoted by $S(t_0, T)$, is $E[S_T]$. From Appendix D.2 we have

$$S(t_0, T) = E[S_T] = S\exp\big(r(T-t_0)\big) \tag{4.4.32}$$

Using Eq. (2.3.9) the distribution of the asset price at time T is:

$$\log\left(\frac{S_T}{S}\right) \sim \mathrm{N}\big(\nu(T-t_0), \sigma^2(T-t_0)\big) \tag{4.4.33}$$

We want to obtain the current price of a vanilla European option with strike price E which matures at future time T, and thus has a duration of $\tau = T - t_0$. The approach we will adopt here is to first derive an expression for the value of a European call option, and then use the put/call parity relationships of Section 2.2 to obtain the value of the corresponding European put option.

Referring to (4.2.1) we have

$$V_{t_0} = B_{t_0} E^{\mathbb{Q}}\left[\frac{H_T}{B_T}\right] = \frac{B_{t_0}}{B_T} E^{\mathbb{Q}}[H_T] \tag{4.4.34}$$

Substituting $B_{t_0} = 1$, $B_T = \exp(r(T-t_0)) = \exp(r\tau)$, and $H_T = \max(S_T - E, 0)$ we have:

$$V_{t_0} = \frac{1}{\exp(r\tau)} E^{\mathbb{Q}}\big[\max(S_T - E, 0)\big] \tag{4.4.35}$$

and so denoting the value of the call option by $c(S, E, \tau)$ we obtain

$$c(S, E, \tau) = \exp(-r\tau) E^{\mathbb{Q}}\big[\max(S_T - E, 0)\big] \tag{4.4.36}$$

It can be seen from Eq. (4.4.36) that the value of the European call option is the expected value of the option's payoff at maturity, discounted to current time t by the riskless interest rate r.

This means that the value of the call option can be written as

$$c(S, E, \tau) = \exp(-r\tau) \int_{S_T=E}^{\infty} f(S_T)(S_T - E)\,\mathrm{d}S_T \tag{4.4.37}$$

where $f(S_T)$ is the probability density function of S_T.

Instead of integrating over S_T we will evaluate (4.4.37) by using the variable $X = \log(S_T/S)$. From Eq. (4.4.33), we know that the probability density function of X is

$$f(X) = \frac{1}{\sigma\sqrt{\tau}\sqrt{2\pi}} \exp\left(-\frac{(X - (r - \sigma^2/2)\tau)^2}{2\sigma^2\tau}\right) \tag{4.4.38}$$

and therefore the value of the option is

$$c(S, E, \tau) = \exp(-r\tau) \int_{X=\log(E/S)}^{\infty} \left\{ S \exp(X) - E \right\} f(X)\, \mathrm{d}X \tag{4.4.39}$$

where we have used $S_T = S\exp(X)$. The lower limit in Eq. (4.4.39) corresponding to $S_T = E$ in Eq. (4.4.37) is found by setting $E = \exp(X)$; this yields the lower limit $X = \log(E/S)$.

The integral in Eq. (4.4.39) is evaluated by splitting it into the two parts:

$$c(S, E, \tau) = I_A - I_B \tag{4.4.40}$$

where

$$I_A = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp(X) \exp\left(-\frac{\{X - (r - \sigma^2/2)\tau\}^2}{2\sigma^2\tau} \right) \mathrm{d}X \tag{4.4.41}$$

and

$$I_B = \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X - (r - \sigma^2/2)\tau\}^2}{2\sigma^2\tau} \right) E\, \mathrm{d}X \tag{4.4.42}$$

To evaluate these integrals we will make use of the fact that the univariate cumulative normal function $N_1(x)$ is:

$$N_1(x) = \frac{1}{\sqrt{2\pi}} \int_{u=-\infty}^{x} \exp\left(-\frac{u^2}{2} \right) \mathrm{d}u$$

By symmetry we have $N_1(-x) = 1 - N_1(x)$ and

$$\frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} \exp\left(-\frac{u^2}{2} \right) \mathrm{d}u = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-x} \exp\left(-\frac{u^2}{2} \right) \mathrm{d}u = N_1(-x)$$

We will first consider I_B, which is the easier of the two integrals.

$$I_B = \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X - (r - \sigma^2/2)\tau\}^2}{2\sigma^2\tau} \right) \mathrm{d}X$$

If we let $u = \frac{X - (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$ then $\mathrm{d}X = \sigma\sqrt{\tau}\, \mathrm{d}u$. So

$$I_B = \frac{E\exp(-r\tau)\sigma\sqrt{\tau}}{\sigma\sqrt{2\pi}\sqrt{\tau}} \int_{u=k_2}^{\infty} \exp\left(-\frac{u^2}{2} \right) \mathrm{d}u$$

where the lower integration limit is $k_2 = \frac{\log(E/S) - (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$.

We therefore have:

$$I_B = E\exp(-r\tau) N_1(-k_2) \tag{4.4.43}$$

We will now consider the integral I_A.

$$I_A = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp(X) \exp\left(-\frac{\{X - (r - \sigma^2/2)\tau\}^2}{2\sigma^2\tau} \right) \mathrm{d}X$$

Rearranging the integrand:

$$I_A = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X-(r-\sigma^2/2)\tau\}^2 - 2\sigma^2\tau X}{2\sigma^2\tau}\right) \mathrm{d}X \tag{4.4.44}$$

Expanding the terms in the exponential:

$$\begin{aligned}
&\left\{X-\left(r-\sigma^2/2\right)\tau\right\}^2 - 2\sigma^2\tau X \\
&\quad = X^2 - 2\left\{\left(r-\sigma^2/2\right)\tau\right\}X + \left\{\left(r-\sigma^2/2\right)\tau\right\}^2 - 2\sigma^2\tau X \\
&\quad = X^2 - 2\left\{\left(r+\sigma^2/2\right)\tau\right\}X + \left\{\left(r-\sigma^2/2\right)\tau\right\}^2 \\
&\quad = \left\{X-\left(r+\sigma^2/2\right)\tau\right\}^2 + \left\{\left(r-\sigma^2/2\right)\tau\right\}^2 - \left\{\left(r+\sigma^2/2\right)\tau\right\}^2
\end{aligned}$$

Which results in:

$$\left\{X-\left(r-\sigma^2/2\right)\tau\right\}^2 - 2\sigma^2\tau X = \left\{X-\left(r+\sigma^2/2\right)\tau\right\}^2 - 2\sigma^2 r\tau^2 \tag{4.4.45}$$

Substituting Eq. (4.4.45) into the integrand of Eq. (4.4.44) we have:

$$\begin{aligned}
&\exp(X)\exp\left(-\frac{\{X-(r-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \\
&\quad = \exp(r\tau)\exp\left(-\frac{\{X-(r+\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)
\end{aligned}$$

The integral I_A can therefore be expressed as:

$$I_A = \frac{S\exp(r\tau)\exp(-r\tau)}{\sigma\tau\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X-(r+\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \mathrm{d}X$$

If we let $u = \frac{X-(r+\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$ then $\mathrm{d}X = \sigma\sqrt{\tau}\,\mathrm{d}u$. So

$$I_A = \frac{S\sigma\sqrt{\tau}}{\sigma\sqrt{2\pi}\sqrt{\tau}} \int_{u=k_1}^{\infty} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u$$

where the lower limit of integration is $k_1 = \frac{\log(E/S)-(r+\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$. We therefore have:

$$I_A = SN_1(-k_1) \tag{4.4.46}$$

Therefore the value of a European call is:

$$c(S,E,\tau) = SN_1(-k_1) - E\exp(-r\tau)N_1(-k_2)$$

which gives the usual form of the Black–Scholes formula for a European call as:

$$c(S,E,\tau) = SN_1(d_1) - E\exp(-r\tau)N_1(d_2) \tag{4.4.47}$$

where

$$\begin{aligned} d_1 &= \frac{\log(S/E) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}} \quad \text{and} \\ d_2 &= \frac{\log(S/E) + (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}} = d_1 - \sigma\sqrt{\tau} \end{aligned} \tag{4.4.48}$$

To gain some insight into the meaning of the above equation, we will rewrite it in the following form:

$$c(S, E, \tau) = \exp(-r\tau)\big\{SN_1(d_1)\exp(r\tau) - EN_1(d_2)\big\} \tag{4.4.49}$$

The term $N_1(d_2)$ is the probability that the option will be exercised in a risk-neutral world, so that $EN_1(d_2)$ is the strike price multiplied by the probability that the strike price will be paid. The term $SN_1(d_1)\exp(r\tau)$ is the expected value of a variable, in a risk neutral world, that equals S_T if $S_T > E$ and is otherwise zero.

The corresponding formula for a put can be shown using put call parity (see Section 4.3) to be:

$$p(S, E, \tau) = E\exp(-r\tau)N_1(-d_2) - SN_1(-d_1) \tag{4.4.50}$$

or equivalently, using $N_1(-x) = 1 - N_1(x)$ we have

$$p(S, E, \tau) = E\exp(-r\tau)\big\{1 - N_1(d_2)\big\} - S\big\{1 - N_1(d_1)\big\} \tag{4.4.51}$$

The inclusion of continuous dividends

The effect of dividends on the value of a European option can be dealt with by assuming that the asset price is the sum of a riskless component involving known dividends that will be paid during the life of the option, and a risky (stochastic) component; see Hull (2003).

As continuous dividends q are paid, the stock price is reduced by the same amount, and by the time the European option matures all the dividends will have been paid, leaving only the risky component of the asset price.

From Eq. (4.4.26) we thus have:

$$\begin{aligned} \mathrm{d}S &= \mu S\,\mathrm{d}t - Sq\,\mathrm{d}t + \sigma S\,\mathrm{d}W^P \\ \mathrm{d}B &= rB\,\mathrm{d}t \end{aligned} \tag{4.4.52}$$

where under probability measure $\mathbb{P}$ we know that $\mathrm{d}W^P \sim \mathrm{N}(0, \mathrm{d}t)$.

As before (using Girsanov's theorem), we choose probability measure $\mathbb{Q}$ so that

$$\mathrm{d}W^p = \mathrm{d}W^Q - \left(\frac{\mu - r}{\sigma}\right)\mathrm{d}t$$

and thus under this measure the process for S is

$$\mathrm{d}S = S\mu\,\mathrm{d}t - Sq\,\mathrm{d}t - \left(\frac{\mu - r}{\sigma}\right)\mathrm{d}t + S\sigma\,\mathrm{d}W^Q, \quad \text{where } \mathrm{d}W^Q \sim \mathrm{N}(0, \mathrm{d}t)$$

which results in

$$\mathrm{d}S = S(r-q)\,\mathrm{d}t + \sigma S\,\mathrm{d}W^{Q} \quad \text{where } \mathrm{d}W^{Q} \sim \mathrm{N}(0, \mathrm{d}t) \tag{4.4.53}$$

Proceeding as before we obtain:

$$X \sim \mathrm{N}\big(\{r - q - \sigma^2/2\}\tau, \sigma^2\tau\big)$$

where $X = S_T/S$. The probability density function of X is now:

$$f(X) = \frac{1}{\sigma\sqrt{\tau}\sqrt{2\pi}} \exp\left(-\frac{(X-(r-q-\sigma^2/2)\tau)^2}{2\sigma^2\tau}\right)$$

The value of a call option is thus:

$$c(S,E,\tau) = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \big\{S\exp(X) - E\big\} \times \exp\left(-\frac{(X-(r-q-\sigma^2/2)\tau)^2}{2\sigma^2\tau}\right)\mathrm{d}X \tag{4.4.54}$$

with

$$I_A = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp(X)\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\mathrm{d}X$$

and

$$I_B = \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)E\,\mathrm{d}X$$

So $I_B = E\exp(-r\tau)N_1(-k_2)$, where $k_2 = \frac{\log(E/S)-(r-q-\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$.

We will now consider the integral I_A.

$$I_A = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp(X)\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\mathrm{d}X$$

Rearranging the integrand:

$$I_A = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2 - 2\sigma^2\tau X}{2\sigma^2\tau}\right)\mathrm{d}X$$

Expanding the exponential, we obtain:

$$\begin{aligned}&\big\{X - \big(r-q-\sigma^2/2\big)\tau\big\}^2 - 2\sigma^2\tau X \\ &\quad = 2\big\{X - \big(r-q+\sigma^2/2\big)\tau\big\}^2 - 2\sigma^2(r-q)\tau^2\end{aligned}$$

The integral I_A can therefore be expressed as:

$$I_A = \frac{S\exp((r-q)\tau)\exp(-r\tau)}{\sigma\tau\sqrt{2\pi}} \times \int_{X=\log(E/S)}^{\infty} \exp\left(-\frac{\{X-(r-q+\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\mathrm{d}X$$

which gives $I_A = S\exp(-q\tau)N_1(-k_1)$ where $k_1 = \frac{\log(E/S)-(r-q+\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$.

The Black–Scholes formula for the value of a European call including continuous dividends is thus:

$$c(S, E, \tau) = S\exp(-q\tau)N_1(d_1) - E\exp(-r\tau)N_1(d_2) \tag{4.4.55}$$

and the corresponding formula for a put can be shown (using put call parity) to be:

$$p(S, E, \tau) = -S\exp(-q\tau)N_1(-d_1) + E\exp(-r\tau)N_1(-d_2) \tag{4.4.56}$$

or equivalently, using $N_1(-x) = 1 - N_1(x)$, we have

$$\begin{aligned} p(S, E, \tau) = {} & E\exp(-r\tau)\big\{1 - N_1(d_2)\big\} \\ & - S\exp(-q\tau)\big\{1 - N_1(d_1)\big\} \end{aligned} \tag{4.4.57}$$

where

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}, \qquad d_2 = \frac{\log(S/E) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

Thus European put/call options with continuous dividends can be priced using Eqs. (4.4.47) and (4.4.50) but with S replaced by $S\exp(-q\tau)$.

These formulae can also be re-expressed in terms of the current equity forward price with maturity T, $S(t, T)$, as follows:

$$c_t = \exp\big(-r(T-t)\big)\big\{S(t, T)N_1(d_1) - EN_1(d_2)\big\} \tag{4.4.58}$$

$$p_t = \exp\big(-r(T-t)\big)\big\{-S(t, T)N_1(-d_1) + EN_1(-d_2)\big\} \tag{4.4.59}$$

where we have used the shortened notation p_t and c_t to denote the current (time t) value of put and call options; the current equity forward price with maturity T is

$$S(t, T) = S\exp\big((r - q)(T - t)\big), \quad t \leqslant T,$$

and

$$d_1 = \frac{\log(S(t, T)/E) + (\sigma^2/2)\tau}{\sigma\sqrt{(T-t)}}, \qquad d_2 = \frac{\log(S(t, T)/E) - (\sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}$$

The inclusion of discrete dividends

Here we consider n discrete cash dividends $D_i, i = 1, \ldots, n$, paid at times $t_i, i = 1, \ldots, n$, during the life of the option. In these circumstances the Black–Scholes formula can be used to price European options, but with the current asset value S reduced by the present value of the cash dividends.

This means that instead of S we use the quantity S_D which is computed as

$$S_D = S - \sum_{i=1}^{n} D_i \exp(-rt_i)$$

where r is the (in this case constant) riskless interest rate. The formulae for European puts and calls is then

$$c(S, E, \tau) = S_D N_1(d_1) - E \exp(-r\tau) N_1(d_2) \tag{4.4.60}$$

$$p(S, E, \tau) = E \exp(-r\tau)\{1 - N_1(d_2)\} - S_D\{1 - N_1(d_1)\} \tag{4.4.61}$$

where

$$\begin{aligned} d_1 &= \frac{\log(S_D/E) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}} \quad \text{and} \\ d_2 &= \frac{\log(S_D/E) + (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}} = d_1 - \sigma\sqrt{\tau} \end{aligned} \tag{4.4.62}$$

The Greeks

Now that we have derived formulae to price European vanilla puts and calls it is possible to work out their partial derivatives (hedge statistics). We will now merely quote expressions for the Greeks (hedge statistics) for European options. Here the subscript c refers to a European call, and the subscript p refers to a European put. Complete derivations of these results can be found in Appendix A.

Gamma:

$$\Gamma_c = \frac{\partial^2 c}{\partial S^2} = \Gamma_p = \frac{\partial^2 p}{\partial S^2} = \exp(-q\tau)\frac{n(d_1)}{S\sigma\sqrt{\tau}} \tag{4.4.63}$$

Delta:

$$\begin{aligned} \Delta_c &= \frac{\partial c}{\partial S} = \exp(-q\tau) N_1(d_1) \\ \Delta_p &= \frac{\partial p}{\partial S} = \exp(-q\tau)\{N_1(d_1) - 1\} \end{aligned} \tag{4.4.64}$$

Theta:

$$\begin{aligned} \Theta_c &= \frac{\partial c}{\partial t} = q \exp(-q\tau) S N_1(d_1) - rE \exp(-r\tau) N_1(d_2) \\ &\qquad - \frac{Sn(d_1)\sigma \exp(-q\tau)}{2\sqrt{\tau}} \\ \Theta_p &= \frac{\partial p}{\partial t} = -q \exp(-q\tau) S N_1(-d_1) + rE \exp(-r\tau) N_1(-d_2) \\ &\qquad - \frac{Sn(d_1)\sigma \exp(-q\tau)}{2\sqrt{\tau}} \end{aligned} \tag{4.4.65}$$

Rho:

$$\rho_c = \frac{\partial c}{\partial r} = E\tau N_1(d_2), \qquad \rho_p = \frac{\partial p}{\partial r} = -E\tau N_1(-d_2) \tag{4.4.66}$$

Vega:

$$\mathcal{V}_c = \frac{\partial c}{\partial \sigma} = \mathcal{V}_p = \frac{\partial p}{\partial \sigma} = S\exp(-q\tau)n(d_1)\sqrt{\tau} \tag{4.4.67}$$

where $n(x) = \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)$

We now present, in Code excerpt 4.1, a computer program to calculate the Black–Scholes option value and Greeks given in Eqs. (4.4.63)–(4.4.67). The routine uses `EPS = 1.0e-16` to identify whether the arguments are too small,

```
void black_scholes(double *value, double greeks[], double s0, double x,
double sigma, double t, double r, double q, long put, long *iflag)
{
/* Input parameters:
   =================
   s0 - the current price of the underlying asset
   x - the strike price
   sigma - the volatility
   t - the time to maturity
   r - the interest rate
   q - the continuous dividend yield
   put - if put is 0 then a call option, otherwise a put option
   Output parameters:
   ==================
   value - the value of the option
   greeks[] - the hedge statistics output as follows: greeks[0] is gamma, greeks[1] is delta
              greeks[2] is theta, greeks[3] is rho, and greeks[4] is vega
   iflag - an error indicator
*/
      double one=1.0,two=2.0,zero=0.0;
      double eps,d1,d2,temp,temp1,temp2,pi,np;

      if( (x < EPS) || (sigma < EPS) || (t < EPS) ) { /* Check if any of the the input
      arguments are too small */
         *iflag = 2;
         return;
      }
      temp = log(s0/x);
      d1 = temp+(r-q+(sigma*sigma/two))*t;
      d1 = d1/(sigma*sqrt(t));
      d2 = d1-sigma*sqrt(t);
      /* evaluate the option price */
      if (put==0)
         *value = (s0*exp(-q*t)*cum_norm(d1)- x*exp(-r*t)*cum_norm(d2));
      else
         *value = (-s0*exp(-q*t)*cum_norm(-d1) + x*exp(-r*t)*cum_norm(-d2));
      if (greeks) { /* then calculate the Greeks */
         temp1 = -d1*d1/two;
         d2 = d1-sigma*sqrt(t);
         np = (one/sqrt(two*PI)) * exp(temp1);
         if (put==0) { /* a call option */
            greeks[1] = (cum_norm(d1))*exp(-q*t); /* delta */
            greeks[2] = -s0*exp(-q*t)*np*sigma/(two*sqrt(t))
              + q*s0*cum_norm(d1)*exp(-q*t)- r*x*exp(-r*t)*cum_norm(d2); /* theta */
            greeks[3] = x*t*exp(-r*t)*cum_norm(d2); /* rho */
         }
         else { /* a put option */
            greeks[1] = (cum_norm(d1) - one)*exp(-q*t); /* delta */
            greeks[2] = -s0*exp(-q*t)*np*sigma/(two*sqrt(t)) -
               q*s0*cum_norm(-d1)*exp(-q*t) + r*x*exp(-r*t)*cum_norm(-d2); /* theta */
            greeks[3] = -x*t*exp(-r*t)*cum_norm(-d2); /* rho */
         }
         greeks[0] = np*exp(-q*t)/(s0*sigma*sqrt(t)); /* gamma */
         greeks[4] = s0*sqrt(t)*np*exp(-q*t); /* vega */
   }
   return;
}
```

Code excerpt 4.1 Function to compute the Black–Scholes value for European options.

`PI = 3.14159`, and also the function `cum_norm` to compute the cumulative normal distribution function.

It can be seen in Tables 4.1 and 4.2 that the values for gamma and vega are the same for both puts and calls. We can also demonstrate that the option values are consistent by using put call parity.

$$c(S, E, \tau) + E\exp(-r\tau) = p(S, E, \tau) + S\exp(-q\tau)$$

For example, when $\tau = 1.0$ we have $c(S, E, \tau) = 12.952$ and $P(S, E, T) = 9.260$. So: $c(S, E, \tau) + E\exp(-r\tau) = 12.952 + 100 \times \exp(-0.1) = 103.436$ and $p(S, E, \tau) + S\exp(-q\tau) = 9.260 + 100 \times \exp(-0.06) = 103.436$.

Table 4.1 European put option values and Greeks

τ	Value	Delta	Gamma	Theta	Vega	Rho
0.100	3.558	−0.462	0.042	−16.533	12.490	−4.971
0.200	4.879	−0.444	0.029	−10.851	17.487	−9.860
0.300	5.824	−0.431	0.024	−8.298	21.204	−14.663
0.400	6.571	−0.419	0.020	−6.758	24.241	−19.377
0.500	7.191	−0.408	0.018	−5.698	26.832	−24.004
0.600	7.720	−0.399	0.016	−4.909	29.100	−28.544
0.700	8.179	−0.390	0.015	−4.292	31.118	−32.997
0.800	8.582	−0.381	0.014	−3.792	32.935	−37.364
0.900	8.940	−0.373	0.013	−3.377	34.585	−41.646
1.000	9.260	−0.366	0.012	−3.025	36.093	−45.843

The parameters are: $S = 100.0$, $E = 100.0$, $r = 0.10$, $\sigma = 0.30$, $q = 0.06$.

Table 4.2 European call option values and Greeks

τ	Value	Delta	Gamma	Theta	Vega	Rho
0.100	3.955	0.532	0.042	−20.469	12.490	4.929
0.200	5.667	0.544	0.029	−14.724	17.487	9.744
0.300	6.996	0.552	0.024	−12.109	21.204	14.451
0.400	8.121	0.558	0.020	−10.508	24.241	19.054
0.500	9.113	0.562	0.018	−9.387	26.832	23.557
0.600	10.007	0.566	0.016	−8.539	29.100	27.962
0.700	10.826	0.569	0.015	−7.863	31.118	32.271
0.800	11.584	0.572	0.014	−7.305	32.935	36.485
0.900	12.290	0.574	0.013	−6.832	34.585	40.608
1.000	12.952	0.576	0.012	−6.422	36.093	44.640

The parameters are: $S = 100.0$, $E = 100.0$, $r = 0.10$, $\sigma = 0.30$, $q = 0.06$.

4.4.4 Historical and implied volatility

Obtaining the best estimate of the volatility parameter, σ, in the Black–Scholes formula is of crucial importance. There are many different approaches to volatility estimation. These include:

- Historical estimation
- Implied volatility

We will now consider both historical and implied volatility estimation.

Historical volatility

In this method we calculate the volatility using $n + 1$ historical asset prices, $S_i, i = 0, \ldots, n$, and we assume that the asset prices are observed at the regular time interval, $\mathrm{d}\tau$. Since the asset prices are assumed to follow GBM the volatility is computed as the standard deviation of the n continuously compounded returns, $u_i, i = 1, \ldots, n$, where

$$S_i = S_{i-1} \exp(u_i)$$

or

$$u_i = \log\left(\frac{S_i}{S_{i-1}}\right)$$

We already know (see Eq. (2.1.10)) that the expected standard deviation of the asset returns over the time interval is $\sigma\sqrt{\mathrm{d}\tau}$. This means that we obtain the following expression for $\hat{\sigma}$, the estimated volatility:

$$\hat{\sigma}\sqrt{\mathrm{d}\tau} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(u_i - \bar{u})^2} \tag{4.4.68}$$

or

$$\hat{\sigma} = \sqrt{\frac{1}{(n-1)\,\mathrm{d}\tau}\sum_{i=1}^{n}(u_i - \bar{u})^2} \tag{4.4.69}$$

It is accepted practice to express all times in years, and so the volatility is the *annualized* standard deviation of the returns. There is also the issue of how to account for non-trading days such as weekends and holidays. For example, let us suppose that the history of assets prices $S_i, i = 0, \ldots, n$, was obtained by recording the price on each *trading* day. One approach is to use $\mathrm{d}\tau = 1/N_{\mathrm{td}}$, where N_{td} is the number of trading days in a year. If we take $N_{\mathrm{td}} = 250$ then Eq. (4.4.69) becomes

$$\hat{\sigma} = \sqrt{\frac{250}{(n-1)}\sum_{i=1}^{n}(u_i - \bar{u})^2} \tag{4.4.70}$$

```
void hist_vol(double *sigma, double *err, double data[], long n, double dt, long *ifail)
{
/* Input parameters:
   =================
   data[]       - the data, which consists of n asset prices
   n            - the number of data points
   dt           - the (constant) time spacing between the data points (in years)
   Output parameters:
   ==================
   sigma        - the computed historical volatility
   err          - the standard error in the volatility estimate sigma
   iflag        - an error indicator
*/

#define DATA(I) data[(I)-1]

  double mean=0.0,sum=0.0;
  double temp,tn;
  long i;

  for(i = 2; i <= n; ++i)
     mean = mean + log(DATA(i))-log(DATA(i-1));
  mean = mean/(double)(n-1);

  for(i = 2; i <= n; ++i) {
     temp = log(DATA(i))-log(DATA(i-1));
     sum = sum + (temp-mean)*(temp-mean);
  }
  sum = sum/(double)(n-2);
  *sigma = sqrt(sum/dt);
  tn = (double)(2*(n-1));
  *err = *sigma/sqrt(tn);
  return;
}
```

Code excerpt 4.2 Function to compute the historical volatility from asset data.

The estimated standard error in $\hat{\sigma}$ is (see for example Hull (2003)) given by

$$\hat{\sigma}_{\text{std}} = \hat{\sigma}\sqrt{\frac{1}{2(n-1)}} \tag{4.4.71}$$

A computer program to perform these calculations is given in Code excerpt 4.2.

Implied volatility

The implied volatility of a European option is the volatility that, when substituted into the Black–Scholes equation, yields the market value quoted for the same option. In general the implied volatility will depend on both the time to expiry of the option and also the ratio of the current asset price to the strike—this is known as the volatility smile. These values are usually stored in a multidimensional implied volatility surface, and the volatility for pricing a given option obtained via multidimensional interpolation.

The routine provided in Code excerpt 4.3 uses Newton's method to calculate the implied volatility for a European option from its market price. We will now ilustrate this technique for a European call option with market value `opt_value`. The implied volatility, σ, is then that value which satisfies:

$$K(\sigma) = c(S, E, \tau, \sigma) - \texttt{opt_value} = 0$$

where $c(S, E, \tau, \sigma)$ represents the value of the European call and the other symbols have their usual meaning.

```
 void implied_volatility(double value, double s0, double x, double sigma[],
                         double t, double r, double q, long put, long *iflag)
{
/* Input parameters:
   ================
   value        - the current value of the option
   s0           - the current price of the underlying asset
   x            - the strike price
   sigma[]      - the input bounds on the volatility: sigma[0], the lower bound and, sigma[1],
                  the upper bound
   t            - the time to maturity
   r            - the interest rate
   q            - the continuous dividend yield
   put          - if put is 0 then a call option, otherwise a put option
   Output parameters:
   ==================
   sigma[]      - the element sigma[0] contains the estimated implied volatility
   iflag        - an error indicator
*/
  double zero=0.0;
  double fx, sig1, sig2;
  double val,tolx;
  double temp,eps,epsqrt,temp1,v1;
  long max_iters, i, ind, ir;
  double greeks[5],c[20],sig,vega;
  long done;

   tolx = eps;
   epsqrt = sqrt(EPS);
   if(put == 0)                                    /* a call option */
      temp1 = MAX(s0*exp(-q*t)-x*exp(-r*t),zero);
   else                                           /* a put option */
      temp1 = MAX(x*exp(-r*t)-s0*exp(-q*t),zero);
   v1 = fabs(value-temp1);
   if (v1 <= epsqrt) {                            /* the volatility is too small */
      *iflag = 3;
      return;
   }
   *iflag = 0;
   i = 0;
   max_iters = 50;
   done = 0;
   sig = sigma[0];                                          /* initial estimate */
   val = value;
   while ((i < max_iters) && (!done)) {                     /* Newton iteration */
     black_scholes(&val,greeks,s0,x,sig,t,r,q,put,iflag); /* compute the Black-Scholes option
                                                             value, val    */
     vega = greeks[4];                                      /* and vega. */
     sig1 = sig - ((val - value)/vega);                     /* compute the new estimate of sigma
                                                             using Newton's method */

     if (tolx > fabs((sig1 - sig)/sig1)) {                  /* check whether the specified
                                                             accuracy has been reached */
        done = 1;
     }
     sig = sig1;                                            /* up date sigma */
     ++i;
   }
   sigma[0] = sig1;                                         /* return the estimate for sigma */
   return;
}
```

Code excerpt 4.3 Function to compute the implied volatility of European options.

From Newton's method we have:

$$\sigma_{i+1} = \sigma_i - \frac{F(\sigma_i)}{F'(\sigma_i)}$$

where

$$F'(\sigma_i) = \frac{\partial F}{\partial \sigma} = \frac{\partial c(S, E, \tau, \sigma)}{\partial \sigma} = \mathcal{V}_c$$

Therefore the iterative procedure is

$$\sigma_{i+1} = \sigma_i - \frac{c(S, E, \tau, \sigma) - \texttt{opt_value}}{\mathcal{V}_c}$$

where σ_0 is the initial estimate, and σ_{i+1} is the improved estimate of the implied volatility based on the ith estimate σ_i. Termination of this iteration occurs when $ABS(\sigma_{i+1} - \sigma_i) < \texttt{tol}$, for a specified tolerance, `tol`.

It can be seen that as $\sigma \to 0,\ d_1 \to \infty,\ d_2 \to \infty$ and, from Eq. (4.4.67), we have $\mathcal{V}_c \to 0$. Under these circumstances Newton's method fails.

The same procedure can be used to compute the implied volatility for a European put, in this case we just replace $c(S, E, \tau, \sigma)$ by $p(S, E, \tau, \sigma)$, the value of a European put; from Eq. (4.4.67) $\mathcal{V}_c = \mathcal{V}_p$.

If the implied volatility of American options is required, the procedure is exactly the same. However, instead of using the Black–Scholes formula to compute both the option value and vega we use a binomial lattice to do this. The use of binomial lattices to obtain option prices and the Greeks is described in Chapter 5.

Code excerpt 4.4 provides a simple test program which illustrates the use of the function `implied_volatility`; the results are presented in Table 4.3.

```
double X,  value, S, sigma[2], sigmat, T, r, q;
long i, ifail, put;

ifail  = 0;
S      = 10.0;
X      = 10.5;
r      = 0.1;
sigmat = 0.1;
q      = 0.04;
put    = 0;
printf ("  Time         option value        implied  volatility  (Error)\n");
for(i = 1;i < 6; ++i) {
    T = (double)i*0.5;
    black_scholes(&value,NULL,S,X,sigmat,T,r,q,put,&flag);
    sigma[0] = 0.05;
    sigma[1] = 1.0;
    implied_volatility(value,S,X,sigma,T,r,q,put,&flag);
    printf("%8.4f        %15.4f              %15.4f (%8.4e) \n",T,value,sigma[0],
                                              fabs(sigmat-sigma[0]));
    sigmat = sigmat + 0.1;
}
```

Code excerpt 4.4 Simple test program for function `implied_volatility`.

Table 4.3 Calculated option values and implied volatilities from Code excerpt 4.4

Time (years)	Option value	True σ	Error in estimated σ
0.5	0.1959	0.1	2.7756×10^{-16}
1.0	0.8158	0.2	2.2204×10^{-16}
1.5	1.5435	0.3	3.8858×10^{-16}
2.0	2.3177	0.4	5.5511×10^{-17}
2.5	3.1033	0.5	1.1102×10^{-16}

4.4.5 Pricing options with Microsoft Excel

In this section we show how the Visual Basic within Excel can be used to create powerful derivative pricing applications based on the Black–Scholes formula. We will explain how Excel's Visual Basic can be used to create an application that prices a selection of simple European put and call options at the press of a button.

In Section 4.4.3 we derived the Black–Scholes formula:

$$c(S, E, \tau) = SN_1(d_1) - \mathrm{e}^{-r\tau} EN_1(d_2)$$

and

$$p(S, E, \tau) = -SN_1(-d_1) + \mathrm{e}^{-r\tau} EN_1(-d_2)$$

where

$$d_1 = \frac{\log(S/E)(r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}} = d_1 - \sigma\sqrt{\tau}$$

where S is the current value of the asset and σ is the volatility of the asset, and $N_1(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-x^2/2}\, \mathrm{d}x$.

The univariate cumulative standard normal distribution, $N_1(x)$, can be evaluated in Excel by using its built-in function NORMDIST. The definition of this function is as follows:

```
NORMDIST(x,mean,standard_dev,cumulative)
```

This function returns the normal cumulative distribution for the specified mean and standard deviation.

Function parameters:

`x`:	is the value for which you want the distribution.
`mean`:	is the arithmetic mean of the distribution.
`standard_dev`:	is the standard deviation of the distribution.
cumulative:	is a logical value that determines the form of the function. If cumulative is `TRUE`, `NORMDIST` returns the cumulative distribution function; if `FALSE`, it returns the probability density function.

If `mean = 0` and `standard_dev = 1`, `NORMDIST` returns the standard normal distribution.

This function can be used to create a Visual Basic function to calculate European option values within Excel, see Code excerpt 4.5.

Once the function has been defined, it can be accessed interactively using the Paste Function facility within Excel as shown in Fig. 4.1.

The function `bs_opt` can also be incorporated into other Visual Basic code within Excel. Code excerpt 4.6 defines the Visual Basic subroutine `MANY_EUROPEANS_Click()`.

```
Function bs_opt(S0 As Double, _
  ByVal X As Double, sigma As Double, T As Double, _
  r As Double, q As Double, ByVal putcall As Long) As Double

' Visual Basic Routine to calculate the value of
' either a European Put or European Call option.
' Author: George Levy

 Dim temp As Double
 Dim d1 As Double
 Dim d2 As Double
 Dim SQT As Double
 Dim value As Double

 temp = Log(S0 / X)
 d1 = temp + (r - q + (sigma * sigma / 2#)) * T
 SQT = Sqr(T)
 d1 = d1 / (sigma * SQT)
 d2 = d1 - sigma * SQT


 If (putcall = 0) Then ' a call option
    value = S0 * Exp(-q * T) * WorksheetFunction.NormDist(d1, 0#, 1#, True) _
            - WorksheetFunction.NormDist(d2, 0#, 1#, True) * X * Exp(-r * T)

 Else ' a put option
    value = -S0 * Exp(-q * T) * WorksheetFunction.NormDist(-d1, 0#, 1#, True) + _
         X * WorksheetFunction.NormDist(-d2, 0#, 1#, True) * Exp(-r * T)

 End If

  bs_opt = value

End Function
```

Code excerpt 4.5 Visual Basic code to price European options using the Black–Scholes formula.

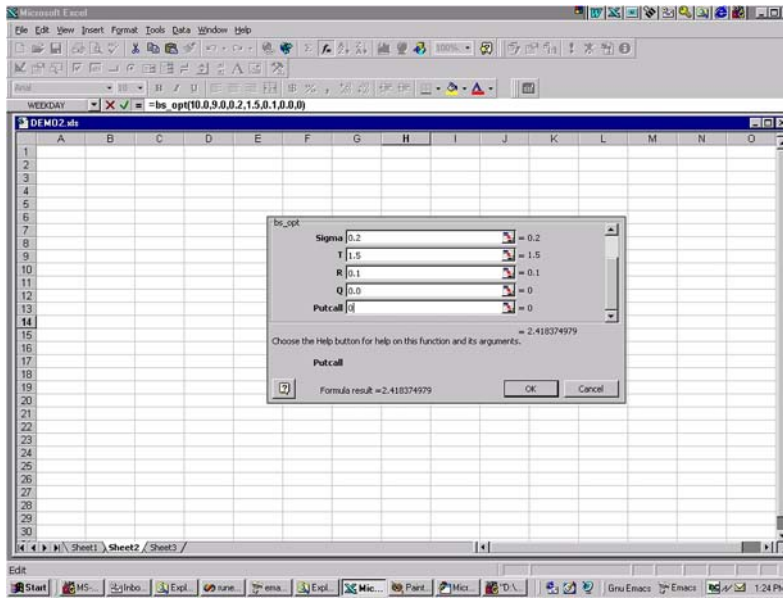

Figure 4.1 Using the function `bs_opt` interactively within Excel. Here a call option is processed with the following parameters: $S = 10.0$, $X = 9.0$, $q = 0.0$, $T = 1.5$, $r = 0.1$, and $\sigma = 0.2$.

```
Private Sub MANY_EUROPEANS_Click()

Dim i As Long
Dim putcall As Long
Dim S0 As Double
Dim q As Double
Dim sigma As Double
Dim T As Double
Dim r As Double

q = 0#
T = 1.5
r = 0.1
sigma = 0.2

For i = 1 To 22

  S0 = Sheet1.Cells(i + 1, 1).value
  X = Sheet1.Cells(i + 1, 2).value
  putcall = Sheet1.Cells(i + 1, 3).value
  Sheet1.Cells(i + 1, 4).value = bs_opt(S0, X, sigma, T, r, q, putcall)

Next i

End Sub
```

Code excerpt 4.6 Visual Basic code that uses the function `bs_opt`.

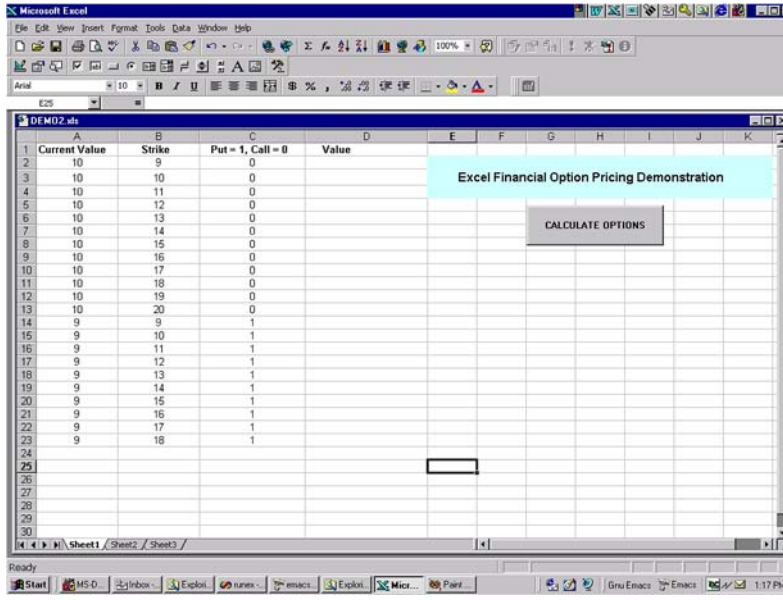

Figure 4.2 Excel worksheet before calculation of the European option values.

When the button labelled “CALCULATE OPTIONS” is clicked, the values of 22 European options will be calculated using the data in columns 1–3 on worksheet 1, see Figs. 4.2 and 4.3.

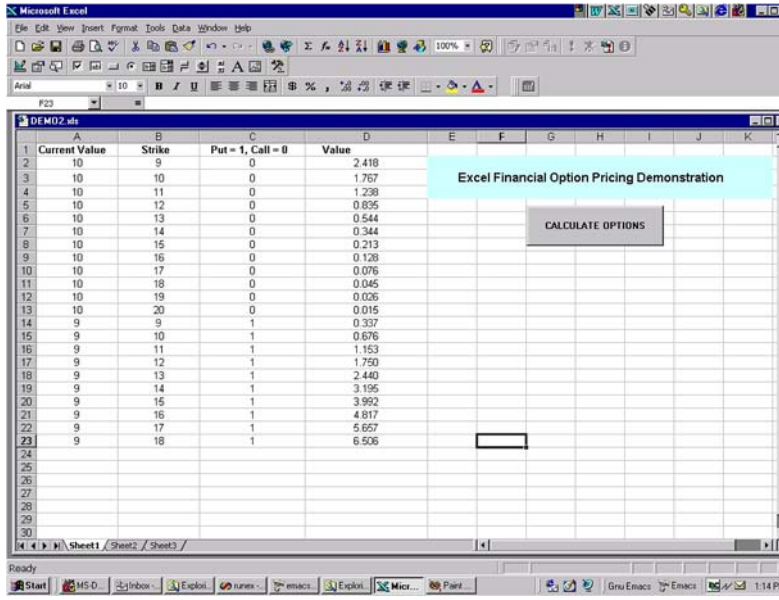

Figure 4.3 Excel worksheet after calculation of the European option values.

The cumulative standard normal distribution can also be used to provide analytic solutions for a range of other *exotic* options such as: Barrier options, Exchange options, Lookback options, Binary options, etc.

4.5 Barrier options

4.5.1 Introduction

Barrier options are derivatives where the payoff depends on whether the asset price reaches a given barrier level, B. *Knockout options* become worthless (cease to exist) if the asset price reaches the barrier, whereas *knockin options* come into existence when the asset price *hits* the barrier. We will consider the following single asset European barrier options:

- *Down and out call*: A knockout vanilla call option, value c_{do}, which ceases to exist when the asset price reaches or goes below the barrier level.
- *Up and out call*: A knockout vanilla call option, value c_{uo}, which ceases to exist when the asset price reaches, or goes above the barrier level.
- *Down and in call*: A knockin vanilla call option, value c_{di}, which comes into existence when the asset prices reaches or goes below the barrier level.

- *Up and in call*: A knockin vanilla call option, value c_{ui}, which comes into existence when the asset price reaches or goes above the barrier level.

The following expressions must be true:

$$c = c_{\text{uo}} + c_{\text{ui}} \tag{4.5.1}$$

$$c = c_{\text{do}} + c_{\text{di}} \tag{4.5.2}$$

where c is the value of a vanilla call option. We thus need only derive expressions for both the knockout options, and then use the above equations to calculate the value of the corresponding knockin options.

The notation that we will use is as follows: The symbol t represents the current time, T represents the time at which the option matures, and $\tau = T - t$, the duration of the option. The symbol s, with constraint $t \leqslant s \leqslant T$, is any intermediate time during which the option is alive.

4.5.2 Analytic pricing of down and out call options

If we consider Brownian motion (with zero drift) $X_s \sim \text{N}(0, (s-t)\sigma^2)$, $t \leqslant s \leqslant T$, which starts at $X_t = 0$ and, after time $\tau = T - t$, ends at the point $X_T = X$ then (for example, see Freedman, 1983) the probability density function for this motion not to exceed the value $X = b$ (where $b > 0$) during time τ is given by:

$$f\left(b \geqslant X_s^{\max}, X\right) = \Omega\sqrt{\frac{2}{\pi}}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\exp\left(-\frac{X^2}{2\sigma^2\tau}\right) \tag{4.5.3}$$

where for convenience we have used $\Omega = (2b - X)/(\sigma^3\tau^{3/2})$, and $X_s^{\max} = \max(X_s, t \leqslant s \leqslant T)$. Since X_s is Brownian motion without drift and volatility σ, then $-X_s$ is identical Brownian motion. Therefore by substituting $X \to -X$, and $b \to -b$ in the above equation we obtain:

$$f\left(b \leqslant X_s^{\min}, X\right) = -\Omega\sqrt{\frac{2}{\pi}}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\exp\left(-\frac{X^2}{2\sigma^2\tau}\right) \tag{4.5.4}$$

where we have used $X_s^{\min} = \min(X_s, t \leqslant s \leqslant T)$. Equation (4.5.4) is the probability density function of $-X_s$ staying above the value $X = b$, where $b < 0$. These results can be generalized to include drift (Musiela and Rutkowski, 1998, p. 212), so that $X_s \sim \text{N}((r - q - \sigma^2/2)(s-t), \sigma(s-t))$, for $t \leqslant s \leqslant T$. We now have the following results:

$$\begin{aligned} &f\left(b \geqslant X_s^{\max}, X\right) \\ &\quad = \Omega\sqrt{\frac{2}{\pi}}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\exp\left(-\frac{(X-(r-q-\sigma^2/2)\tau)^2}{2\sigma^2\tau}\right) \end{aligned} \tag{4.5.5}$$

$$f\left(b \leqslant X_s^{\min}, X\right) = -\Omega\sqrt{\frac{2}{\pi}}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\exp\left(-\frac{(X-(r-q-\sigma^2/2)\tau)^2}{2\sigma^2\tau}\right) \quad (4.5.6)$$

where r is the risk free rate and q is the continuous dividend yield. A European down and out barrier option with maturity τ and a barrier at $X = B$ will cease to exist (become worthless) if at any time $X_s \leqslant B$, for $t \leqslant s \leqslant T$. The probability density function that the barrier option will continue to exist at time T if the end point is X is therefore:

$$f(X > B) = -\sqrt{\frac{2}{\pi}}\int_{B=S\exp(b)}^{b=X}\Omega\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \times \exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\mathrm{d}b \quad (4.5.7)$$

or

$$f(X > B) = -\sqrt{\frac{2}{\pi}}\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \int_{b=\log(B/S)}^{b=X}\Omega\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\mathrm{d}b \quad (4.5.8)$$

where we have integrated over all possible values of b (i.e., $B < b < X$) that keep the option alive. Recalling that:

$$-\int_{b=\log(B/S)}^{b=X}\Omega\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\mathrm{d}b = \int_{b=\log(B/S)}^{b=X}\frac{(X-2b)}{\sigma^3\tau^{3/2}}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\mathrm{d}b$$

and noting that:

$$\frac{\partial}{\partial b}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) = \frac{2(X-2b)}{\sigma^2\tau}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)$$

we have:

$$\int_{b=\log(B/S)}^{b=X}\frac{2(X-2b)}{\sigma^2\tau}\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\mathrm{d}b = \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\Bigg]_{b=\log(B/S)}^{b=X} = 1-\exp\left(\frac{2\log(B/S)(X-\log(B/S))}{\sigma^2\tau}\right)$$

So we have:

$$f(X > B) = \frac{1}{\sigma\sqrt{\tau}\sqrt{2\pi}}\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \left\{1-\exp\left(\frac{2\log(B/S)(X-\log(B/S))}{\sigma^2\tau}\right)\right\}$$

The value c_{do} of a European down and out call option with strike E, satisfying $E > B$, is given by:

$$c_{\text{do}} = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \{S\exp(X) - E\} f(X > B)\,\mathrm{d}X \tag{4.5.9}$$

This integral is evaluated in Appendix B.1, and the value of the down and out call option c_{do} is:

$$c_{\text{do}} = c - c_{\text{di}} \tag{4.5.10}$$

where

$$c = S\exp(-q\tau)N_1(d_1) - E\exp(-r\tau)N_1(d_2)$$

$$c_{\text{di}} = S\exp(-q\tau)N_1(d_4)\left(\frac{B}{S}\right)^{\frac{2(r-q)}{\sigma^2}+1} - E\exp(-r\tau)N_1(d_3)\left(\frac{B}{S}\right)^{\frac{2(r-q)}{\sigma^2}-1}$$

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_2 = \frac{\log(S/E) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_3 = \frac{\log(B^2/SE) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$d_4 = \frac{\log(B^2/ES) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

In Code excerpt 4.7 we provide the function `bs_opt_barrier_downout_call` which uses Eq. (4.5.10) to price a down and out European call option. This routine will be used in Chapter 5 to measure the accuracy achieved by using various finite-difference grid techniques to solve the Black–Scholes equation.

4.5.3 Analytic pricing of up and out call options

Here we will obtain an expression for an *up and out* European call option with continuous dividend yield q, in a similar manner to that used in Section 4.5.2 for the down and out European call option. A European up and out barrier option with maturity τ and a barrier at $X = B$ will cease to exist (become worthless) if at any time $X_s \geqslant B$, for $t \leqslant s \leqslant T$. The probability density function that the barrier option will continue to exist at time T if the end point is X is therefore:

$$f(X < B) = \sqrt{\frac{2}{\pi}} \int_{b=X}^{B=S\exp(b)} \Omega \exp\left(\frac{2b(X-b)}{\sigma^2 t}\right) \times \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \mathrm{d}b \tag{4.5.11}$$

```
void bs_opt_barrier_downout_call(double *value, double barrier_level,
           double s0, double x, double sigma, double t, double r,
           double q, long *iflag)
{
/* Input parameters:
   ================
   barrier_level         - the level of the barrier
   s0                    - the current price of the underlying asset
   x                     - the strike price
   sigma                 - the volatility
   t                     - the time to maturity
   r                     - the risk free interest rate
   q                     - the dividend yield
   Output parameters:
   ==================
   value                 - the value of the option
   iflag                 - an error indicator
*/
    double one=1.0,two=2.0,zero=0.0;
    double temp,temp1,temp2,a,b,d1,d2,d3,d4,d5,d6,d7,d8;
    double fac;

      if(x < EPS) { /* then strike price (X) is too small */
         *iflag = 2;
         return;
      }
      if (sigma < EPS) { /* then volatility (sigma) is too small */
         *iflag = 3;
         return;
      }
      if (t < EPS) { /* then time to expiry (t) is too small */
         *ifail = 3;
         return;
      }
      if (barrier_level <= 0) {  /* barrier level must be greater than zero */
         *iflag = 4;
      }

      if (s0 <= barrier_level) { /* option has already been knocked out */
         *value = 0.0;
         return;
      }

      fac = sigma*sqrt(t);
      temp1 = -one+(two*(r-q)/(sigma*sigma));
      temp2 = barrier_level/s0;
      a = pow(temp2,temp1);
      temp1 = one+(two*(r-q)/(sigma*sigma));
      b = pow(temp2,temp1);
      if (x > barrier_level) {   /* strike > barrier_level */
         d1 = (log(s0/x)+(r-q+0.5*sigma*sigma)*t)/fac;
         d2 = (log(s0/x)+(r-q-0.5*sigma*sigma)*t)/fac;
         temp = (s0*x)/(barrier_level*barrier_level);
         d7 = (log(temp)-(r-q-0.5*sigma*sigma)*t)/fac;
         d8 = (log(temp)-(r-q+0.5*sigma*sigma)*t)/fac;

         temp1 = s0*exp(-q*t)*(cum_norm(d1)-b*(one-cum_norm(d8)));
         temp2 = x*exp(-r*t)*(cum_norm(d2)-a*(one-cum_norm(d7)));
         *value = temp1-temp2;
      }
      else { /* strike <= barrier_level */
         d3 = (log(s0/barrier_level)+(r-q-0.5*sigma*sigma)*t)/fac;
         d6 = (log(s0/barrier_level)-(r-q-0.5*sigma*sigma)*t)/fac;
         d4 = (log(s0/barrier_level)+(r-q+0.5*sigma*sigma)*t)/fac;
         d5 = (log(s0/barrier_level)-(r-q+0.5*sigma*sigma)*t)/fac;

         temp1 = s0*exp(-q*t)*(cum_norm(d3)-b*(one-cum_norm(d6)));
         temp2 = x*exp(-r*t)*(cum_norm(d4)-a*(one-cum_norm(d5)));
         *value = temp1-temp2;
      }
     return;
}
```

Code excerpt 4.7 Function to compute the value for European down and out call options.

or

$$f(X < B) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \int_{b=X}^{b=\log(B/S)} \Omega \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \mathrm{d}b \tag{4.5.12}$$

where, as in Section 4.5.2, we have used $\Omega = \frac{(2b-X)}{\sigma^3\tau^{3/2}}$ and have integrated over all possible values of b (i.e., $B > b > X$) that keep the option alive. Recalling that:

$$\int_{b=X}^{b=\log(B/S)} \Omega \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \mathrm{d}b = \int_{b=X}^{b=\log(B/S)} \frac{(2b-X)}{\sigma^3\tau^{3/2}} \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \mathrm{d}b$$

and noting:

$$-\frac{\partial}{\partial b} \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) = \frac{2(X-2b)}{\sigma^2\tau} \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \tag{4.5.13}$$

we have:

$$\begin{aligned}\int_{b=X}^{b=\log(B/S)} \frac{2(2b-X)}{\sigma^2\tau} \exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right) \mathrm{d}b &= -\exp\left(\frac{2b(X-b)}{\sigma^2\tau}\right)\Bigg]_{b=X}^{b=\log(B/S)} \\ &= \left\{1 - \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right)\right\}\end{aligned}$$

Therefore:

$$f(X < B) = \frac{1}{\sigma\sqrt{\tau}\sqrt{2\pi}} \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \left\{1 - \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right)\right\} \tag{4.5.14}$$

We will now derive the formula for an up and out call option when $E < B$. In fact if $E > B$ then the option is worthless, since at the current time t the call option's payout, $\max(S_t - E, 0) = 0$, and if $S_t > E$ then the option will be knocked out.

$$c_{\mathrm{uo}} = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \left\{S\exp(X) - E\right\} f(X < B)\, \mathrm{d}X \tag{4.5.15}$$

Taking into account the fact the option becomes worthless when $S\exp(X) > B$, (i.e., $X > \log(B/S)$) we have:

$$c_{\mathrm{uo}} = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \left\{S\exp(X) - E\right\} f(X < B)\, \mathrm{d}X \tag{4.5.16}$$

This integral is evaluated in Appendix B.2, and the value of the down and out call option c_{uo} is:

$$c_{\text{uo}} = c - c_{\text{ui}}$$

where c is the value of a vanilla call and c_{ui}, the value of an up and in call, is given by:

$$\begin{aligned} c_{\text{ui}} &= S\exp(-q\tau)N_1(d_2) - E\exp(-r\tau)N_1(d_4) \\ &\quad - E\exp(-r\tau)\big\{N_1(d_5) - N_1(d_6)\big\}\left(\frac{B}{S}\right)^{\frac{2(r-q)}{\sigma^2}-1} \\ &\quad + S\exp(-r\tau)\big\{N_1(d_7) - N_1(d_8)\big\}\left(\frac{B}{S}\right)^{\frac{2(r-q)}{\sigma^2}+1} \end{aligned} \tag{4.5.17}$$

and

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_2 = \frac{\log(S/B) + (r - q + \sigma^2/2)\tau}{\sqrt{\tau}}$$

$$d_3 = \frac{\log(S/E) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_4 = \frac{\log(S/B) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_5 = \frac{\log(B^2/ES) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_6 = \frac{\log(B/S) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_7 = \frac{\log(B^2/ES) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_8 = \frac{\log(B/S) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

4.5.4 Monte Carlo pricing of down and out options

In this section we show how Monte Carlo simulation can be used to price down and out barrier options. We will describe both a *basic* Monte Carlo approach and also a Brownian bridge method which gives more accurate results (see Chapter 8).

The asset price, S, will be assumed to be GBM, so the logarithm of the asset price X follows the Brownian process:

$$\Delta X = \nu\Delta t + \sigma\Delta W_t \tag{4.5.18}$$

where ν is the drift and σ is the volatility.

If the barrier level is B then the option will be *knocked out* when $S \leqslant B$, or equivalently $\log(S) \leqslant \log(B)$. This will be expressed as $X \leqslant b$, where $b = \log(B)$.

The basic approach to simulating the down and out option is to first decide how many `Scenarios` to use and also how many `TimeSteps` there are to be in each scenario. The size of each time step is then `time_step = TimeToExpiry/TimeSteps`. For each scenario the path of X_t is advanced in time from t to $t + \Delta t$ using Eq. (4.5.18), with $\mathrm{d}t =$ `time_step` and a value for ΔW_t output from a Gaussian random number generator. Path construction is stopped if either the option expiry time is reached or if the option is knocked out—i.e., $X \leqslant b$. When the option is knocked out before expiry the payoff for that scenario is zero. We will denote the option value obtained from the ith scenario by DO_i where $i = 1, \ldots,$ `Scenarios`. The option value is the average value of DO_i over all scenarios; for more details see Code excerpt 4.8.

One problem with this approach to simulation is that it does not take into account the possibility that $X_\tau \leqslant b$, $t < \tau < t + \Delta t$, even though $X_t > b$ and $X_{t+\Delta t} > b$. In these circumstances the option should be treated as knocked out, since X hit (or crossed) the barrier b at time τ, but then increased to the value $X_{t+\Delta t} > b$ at time $t + \Delta t$.

We will now discuss how the Brownian bridge method deals with this situation.

Let us take two consecutive time points t_1 and $t_2 = t_1 + \Delta t$, and assume that both X_{t_1}, and X_{t_2} are above the (logarithmic) barrier level b. We want to find the probability that in the time interval $[t_1, t_2]$, the asset price went lower than B, and use this to get more accurate values for down and out options. The required barrier crossing probability is thus:

$$P\left(m^X_{t_1,t_2} \leqslant b | \{X_{t_2}, X_{t_1}\}\right)$$

where $m^X_{t_1,t_2}$ denotes the minimum of X over the time interval $[t_1, t_2]$.

The probability density of X_{t_2} conditional on X_{t_1} is

$$p(X_{t_2}|X_{t_1}) = \frac{1}{\sigma\sqrt{2\pi\Delta t}} \exp\left\{-\frac{(X_{t_2} - X_{t_1} - \nu\Delta t)^2}{2\sigma^2\Delta t}\right\}$$

where $\Delta t = t_2 - t_1$.

From Bayes law we know that:

$$P\left(m^X_{t_1,t_2} \leqslant b | \{X_{t_2}, X_{t_1}\}\right) = \frac{p(\{m^X \leqslant b, X_{t_2}\}|X_{t_1})}{p(X_{t_2}|X_{t_1})}$$

We show in Appendix I that

$$p\left(\left\{m^X_{t_1,t_2} \leqslant b, X_{t_2}\right\}|X_{t_1}\right) = \frac{1}{\sigma\sqrt{2\pi\Delta t}} \exp\left\{\frac{2\nu(b - X_{t_1})}{\sigma^2}\right\} \exp\left\{-\frac{(X_{t_2} + X_{t_1} - 2b - \nu\Delta t)^2}{2\sigma^2\Delta t}\right\}$$

so

$$P\left(m^X_{t_1,t_2} \leqslant b | \{X_{t_2}, X_{t_1}\}\right) = \exp\left\{\frac{2\nu(b - X_{t_1})}{\sigma^2}\right\} \times \exp\left\{-\frac{-(X_{t_2} + X_{t_1} - 2b - \nu\Delta t)^2 + (X_{t_2} - X_{t_1} - \nu\Delta t)^2}{2\sigma^2\Delta t}\right\}$$

We will now use some algebra to simplify this expression.

$$\begin{aligned} P\left(m^X_{t_1,t_2} \leqslant b | \{X_{t_2}, X_{t_1}\}\right) &= \exp\left\{\frac{4\nu\Delta t(b - X_{t_1}) - (X_{t_2} + X_{t_1} - 2b - \nu\Delta t)^2 + (X_{t_2} - X_{t_1} - \nu\Delta t)^2}{2\sigma^2\Delta t}\right\} \\ &= \exp\left\{\left(4\nu\Delta t(b - X_{t_1}) - \left((X_{t_2} - X_{t_1} - \nu\Delta t) - 2(b - X_{t_1})\right)^2 \right.\right. \\ &\qquad \left.\left. + (X_{t_2} - X_{t_1} - \nu\Delta t)^2\right)/\left(2\sigma^2\Delta t\right)\right\} \\ &= \exp\left\{\frac{4\nu\Delta t(b - X_{t_1}) + 4(b - X_{t_1})(X_{t_2} - X_{t_1} - \nu\Delta t) + 4(b - X_{t_1})^2}{2\sigma^2\Delta t}\right\} \\ &= \exp\left\{-\frac{2(b - X_{t_1})(b - X_{t_2})}{\sigma^2\Delta t}\right\} \end{aligned}$$

which finally yields

$$P\left(m^{\log(S)}_{t_1,t_2} \leqslant b | \left\{\log(S_{t_2}), \log(S_{t_1})\right\}\right) = \exp\left\{-\frac{2(\log(B) - \log(S_{t_1}))(\log(B) - \log(S_{t_2}))}{\sigma^2\Delta t}\right\} \tag{4.5.19}$$

Equation (4.5.19) gives the probability of the option having been knocked out between times t_1 and t_2 even though the asset prices S_{t_1} and S_{t_2} are greater than B. The probability that the option hasn't been knocked out between times t_1 and t_2 is therefore

$$P\left(m^{\log(S)}_{t_1,t_2} > b | \left\{\log(S_{t_2}), \log(S_{t_1})\right\}\right) = 1 - \exp\left\{-\frac{2(\log(B) - \log(S_{t_1}))(\log(B) - \log(S_{t_2}))}{\sigma^2\Delta t}\right\} \tag{4.5.20}$$

This means that for the (complete) ith scenario path, of n time steps, the probability that $m^{\log(S)} > b$ is

$$BB^i_c = \prod_{j=0}^{n-1}\left\{1 - \exp\left\{-\frac{2(\log(B) - \log(S^i_{t_j}))(\log(B) - \log(S^i_{t_{j+1}}))}{\sigma^2\Delta t}\right\}\right\}$$

where $S^i_{t_j}$ is the ith scenario asset price at time t_j.

The basic Monte Carlo ith scenario option value DO_i can therefore be adjusted as follows

$$DO^*_i = DO_i \; BB^i_c$$

and the new Monte Carlo estimate DO^* is

$$DO^* = \frac{\sum_{i=1}^{\texttt{Scenarios}} DO_i^*}{\texttt{Scenarios}}$$

where more details can be found in Code excerpt 4.8.

```
private double MonteCarloSim(bool is_put)
{
    int seed = 111;
    double[] asset_path = new double[fTimeSteps];
    double time_step = fTimeToExpiry / fTimeSteps;
    double sqrt_time_step = System.Math.Sqrt(time_step);
    double disc = System.Math.Exp(-fRiskFreeRate * fTimeToExpiry);

    set_seed(seed);

    double opt_val = 0.0;
    bool not_out = true;
    int k = 0;
    double STN = 0.0;
    double mean = (fRiskFreeRate - fDividendYield - fSigma1 * fSigma1 * 0.5) * time_step;
    double std = System.Math.Sqrt(fSigma1 * fSigma1 * time_step);
    double z;
    double sum_opt_vals = 0.0;

    for (int i = 0; i < fNumberScenarios; ++i)
    {
        // generate the asset path
        double ST1 = fS1;
        not_out = true;
        k = 0;

        while (not_out && k < fTimeSteps)
        {
            z = RndNorm(mean, std);
            STN = ST1 * System.Math.Exp(z);
            if (STN < fBarrierLevel) not_out = false;
            ST1 = STN;
            asset_path[k] = STN;
            ++k;
        }
        if (is_put)
        {
            opt_val = System.Math.Max(fStrike - STN, 0.0);
        }
        else
        {
            opt_val = System.Math.Max(STN - fStrike, 0.0);
        }
        if (not_out)
        { // only has value if asset value is above the barrier_level
            // compute the probability that the asset remained above the barrier
            if (UseBrownianBridge)
            {
                double total_probability_above = 1.0, pr;
                double sigma_2 = fSigma1 * fSigma1;
                double log_barrier_level = System.Math.Log(fBarrierLevel);
                double fac;
                for (int jj = 0; jj < fTimeSteps - 1; ++jj)
                {
                    double log_S_i = System.Math.Log(asset_path[jj]);
                    double log_S_i1 = System.Math.Log(asset_path[jj + 1]);

                    fac = 2.0 * (log_barrier_level - log_S_i)
                          * (log_barrier_level - log_S_i1) / (sigma_2 * time_step);
                    pr = (1.0 - System.Math.Exp(-fac)); // probability of staying above the
                                                        barrier between i and i+1
                    total_probability_above *= pr;
```

Code excerpt 4.8 An example of using the Brownian bridge barrier crossing probability to enhance the pricing of a European down and out option.

```
                }
                sum_opt_vals += total_probability_above * opt_val * disc;                    }
            else
            { // don't use the Brownian Bridge
                sum_opt_vals += opt_val * disc;
            }
        }
    }
    double temp = sum_opt_vals / (double)fNumberScenarios;

    return temp;
}
```

Code excerpt 4.8 (*Continued*).

This page intentionally left blank

5 Single asset American options

5.1 Introduction

In Chapter 4 we discussed single asset European options and the analytic formulae that can be used to price them. Here we will consider the valuation of single asset American-style options using both numeric methods and analytic formulae; in addition we will discuss the use of numerical techniques to value certain European options. The coverage in this chapter is as follows:

- Analytic approximation techniques for the valuation of American options
- Binomial lattice techniques used for the valuation of American and European options
- The valuation of American and European vanilla and barrier options using finite-difference grids
- The valuation of American options via Monte Carlo simulation.

It should be mentioned that although much of the discussion here concerns the valuation of vanilla European and American puts and calls, the techniques used can be modified without much difficulty to include more exotic options with customized payoffs and early exercise features.

5.2 Approximations for vanilla American options

5.2.1 American call options with cash dividends

In this section we will consider the valuation of vanilla American call options with cash dividends and discuss the methods of Roll, Geske, Whaley and Black. We will first consider the Roll–Geske–Whaley method.

The Roll–Geske–Whaley approximation

This method uses the work of Roll (1977), Geske (1979), and Whaley (1981). Let S be the current (time t) price of an asset which pays a single cash dividend D_1 at time t_1. At the *ex-dividend* date, t_1, there will be a decrease in the asset's value from S_{t_1} to $S_{t_1} - D_1$. Also the current asset price net of *escrowed* dividends is:

$$S_D = S - D_1 \exp\big(-r(t_1 - t)\big) \tag{5.2.1}$$

where r is the riskless interest rate.

Now consider an American call option, with strike price E and expiry time T, which is taken out on this asset. At t_1 there will be a given ex-dividend asset price, S^*, above which the option will be exercised early. This value can be found by solving the following equation:

$$c(S^*, E, \tau_1) = S^* + D_1 - E \tag{5.2.2}$$

where $c(S^*, E, \tau_1)$ is the Black–Scholes value of a European call option with strike price E and maturity $\tau_1 = T - t_1$, on an asset with current value S^* at time t_1. If just prior to the ex-dividend date $S_{t_1} > S^*$ then the American option will be exercised and realize a cash payoff of $S_{t_1} + D_1 - E$. On the other hand if $S_{t_1} \leqslant S^*$ then the option is worth more unexercised and it will be held until option maturity at time T.

We can rewrite Eq. (5.2.2) so that S^* is the root of the following equation:

$$K(S^*) = c(S^*, E, \tau_1) - S^* - D_1 + E = 0 \tag{5.2.3}$$

where $K(S^*)$ denotes the function in the single variable S^*.

A well-known technique for solving Eq. (5.2.3) is Newton's method, which in this case takes the form:

$$S^*_{i+1} = S^*_i - \frac{K(S^*_i)}{K'(S_i)^*} \tag{5.2.4}$$

where S^*_i is the ith approximation to S^* and S^*_{i+1} is the improved $(i + 1)$th approximation.

If we now consider the terms in Eq. (5.2.4) we have from Eqs. (5.2.2) and (5.2.3) that

$$K(S^*_i) = c(S^*_i, E, \tau_1) - S^*_i - D_1 + E$$

and

$$K'(S^*_i) = \frac{\partial K(S^*_i)}{\partial S^*_i} = \frac{\partial c(S^*_i, E, \tau_1)}{\partial S^*_i} - 1$$

Also from Eq. (A.3.2) in Appendix A.3:

$$\frac{\partial c(S^*_i, E, \tau_1)}{\partial S^*_i} = N_1(d_1(S^*_i))$$

we note that here the *continuous* dividend yield $q = 0$.

So

$$K'(S^*_i) = \frac{\partial K(S^*_i)}{\partial S^*_i} = N_1(d_1(S^*_i)) - 1$$

$$\text{where } d_1 = \frac{\log(S^*_i/E) + (r + \sigma^2/2)\tau_1}{\sigma\sqrt{T - t_1}}$$

Substituting these results into Eq. (5.2.4) gives:

$$S^*_{i+1} = S^*_i - \frac{c(S^*_i, E, \tau_1) - (S^*_i + D_1 - E)}{N_1(d_1(S^*_i)) - 1}$$

On rearrangement this yields:

$$S_{i+1}^* = \frac{S_i^* N_1(d_1(S_i^*)) - c(S_i^*, E, \tau_1) + D_1 - E}{N_1(d_1(S_i^*)) - 1}$$
$$\text{for } i = 0, \ldots, \texttt{max_iter} \tag{5.2.5}$$

where a convenient initial approximation is to choose $S_0^* = E$, and `max_iter` is the maximum number of iterations that are to be used.

We will now quote the Roll, Geske, and Whaley formula for the current value of an American call which pays a *single* cash dividend D_1 at time t_1; it is:

$$\begin{aligned} C(S, E, \tau) &= S_D \left\{ N_1(b_1) + N_2\left(a_1, -b_1, \sqrt{\frac{t_1 - t}{\tau}}\right) \right\} \\ &\quad + D_1 \exp\big(-r(t_1 - t)\big) N_1(b_2) \\ &\quad - E \exp(-r\tau) \left\{ N_1(b_2) \exp(r\tau_1) + N_2\left(a_2, -b_2, -\sqrt{\frac{t_1 - t}{\tau}}\right) \right\} \end{aligned} \tag{5.2.6}$$

where S_D is given by Eq. (5.2.1), E is the exercise price, T is the option expiry date, t represents the current time, τ is the option maturity, $N_1(a)$ is the univariate cumulative normal density function with upper intergral limit a, and $N_2(a, b, \rho)$ is the bivariate cumulative normal density function with upper integral limits a and b and correlation coefficient ρ. The other symbols used in Eq. (5.2.6) are defined as

$$a_1 = \frac{\log(S/E) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}, \qquad a_2 = a_1 - \sigma\sqrt{\tau}$$

$$b_2 = \frac{\log(S/S^*) + (r + \sigma^2/2)(t_1 - t)}{\sigma\sqrt{t_1 - t}}, \qquad b_2 = b_1 - \sigma\sqrt{t_1 - t}$$

and S is the current (time t) asset price, S^* is found using Eq. (5.2.5), r is the riskless interest rate, σ is the asset's volatility, $\tau = T - t$ and $\tau_1 = T - t_1$.

To compute the value of an American call option which pays n cash dividends D_i, $i = 1, \ldots, n$, at times t_i, $i = 1, \ldots, n$, we can use the fact that optimal exercise normally only ever occurs at the final ex-dividend date t_n; see for example Hull (2003). Under these circumstances Eq. (5.2.6) can still be shown to value the American call but now t_1 should be set to t_n, D_1 should be set to D_n, and S_D is given by:

$$S_D = S - \sum_{i=1}^{n} D_i \exp\big(-r(t_i - t)\big) \tag{5.2.7}$$

A program to compute the Roll–Geske–Whaley approximation for an American call option with multiple cash dividends is given in Code excerpt 5.1. Here the functions `cum_norm` and `cum_norm2` are used to calculate the values of $N_1(a)$ and $N_2(a, b, \rho)$, respectively. Code excerpt 5.3 was used to compute the values

```
void RGW_approx(double *opt_value, double *critical_value, long n_divs, double dividends[],_
                double Divs_T[],
                double S0, double X, double sigma, double T, double r, long *iflag)
{
/* Input parameters:
   =================
   n_divs         - the number of dividends
   dividends[]    - the dividends: dividends[0] contains the first dividend, dividend[1]
                    the second etc.
   Divs_T[]       - the times at which the dividends are paid: Divs_T[0] is the time at which_
                    the first dividend is paid
                    Divs_T[1] is the time at which the second dividend is paid, etc.
   S0             - the current value of the underlying asset
   X              - the strike price
   sigma          - the volatility
   T              - the time to maturity
   r              - the interest rate
   Output parameters:
   ==================
   opt_value      - the value of the option
   critical_value - the critical value
   iflag          - an error indicator
*/
  double A_1,A_2,S_star,a1,a2,nt1,t1,S;
  double b1,b2,d1,alpha,h,div,beta,temp,temp1,temp2,temp3;
  double pdf,b,eur_val,fac,tol,loc_q,err,zero=0.0;
  long iterate;
  long i,iflagx,putx;

  loc_q = 0.0;
  temp = 0.0;
  for (i=0; i < n_divs; ++i) { /Check the Divs_T array */
    if ((Divs_T[i] <= temp) || (Divs_T[i] > T) || (Divs_T[i] <= zero)) {
       *flag = 2;
       return;
    }
    temp = Divs_T[i];
  }
  /* calculate the present value of the dividends (excluding the final one) */
  temp = 0.0;
  for (i=0; i < n_divs-1; ++i) {
     temp = fac + dividends[i] * exp(-r*Divs_T[i]);
  }
  t1 = Divs_T[n_divs-1];
  /* decrease the stock price by the present value of all dividends */
  div = dividends[n_divs-1];
  S = S0-temp-div*exp(-r*t1);
  iterate = 1;
  tol = 0.000001;
  S_star = X;
  while (iterate) {   /* calculate S_star, iteratively */
      /* calculate the Black-Scholes value of a European call */
      d1 = (log(S_star/X) + (r+(sigma*sigma/2.0))*(T-t1))/(sigma*sqrt(T-t1));
      putx = 0;
      loc_q = 0.0;
      black_scholes(&eur_val,NULL,S_star,X,sigma,T-t1,r,loc_q,putx,&iflag);
      S_star = (S_star*cum_norm(d1)-eur_val+div-X)/(cum_norm(d1)-1.0);
      err = fabs(eur_val - (S_star + div- X))/X;
      if (err < tol) iterate = 0;
  }
  a1 = (log(S/X) + (r+(sigma*sigma/2.0))*T)/(sigma*sqrt(T));
  a2 = a1 - sigma*sqrt(T);
  b1 = (log(S/S_star)+(r+(sigma*sigma/2.0))*t1)/(sigma*sqrt(t1));
  b2 = b1 - sigma*sqrt(t1);
  nt1 = sqrt(t1/T);
  temp1 = S*(cum_norm(b1)+cum_norm2(a1,-b1,-nt1,&iflagx));
  temp2 = -X*exp(-r*T)*cum_norm2(a2,-b2,-nt1,&iflagx)-(X-div)*exp(-r*t1)*cum_norm(b2);
  *opt_value = temp1+temp2;
  *critical_value = S_star;
}
```

Code excerpt 5.1 Function to compute the Roll–Geske–Whaley approximation for the value of an American call option with discrete dividends.

Table 5.1 A comparison of the computed values for American call options with dividends, using the Roll–Geske–Whaley approximation and the Black approximation

Stock price	Critical price, S^*	RGW approximation	Black approximation
80.0	123.582	3.212	3.208
85.0	123.582	4.818	4.808
90.0	123.582	6.839	6.820
95.0	123.582	9.276	9.239
100.0	123.582	12.111	12.048
105.0	123.582	15.316	15.215
110.0	123.582	18.851	18.703
115.0	123.582	22.676	22.470
120.0	123.582	26.748	26.476

The parameters used were: $E = 100.0, r = 0.04, \sigma = 0.2, \tau = 2.0$, and there is one cash dividend of value 5.0 at time $t = 1.0$. The current stock price, S, is varied from 80.0 to 120.0. The results are in agreement with those given in Table 1 of Whaley (1981).

presented in Table 5.1. These compare the Roll–Geske–Whaley approximation with the Black approximation, which we will now briefly discuss.

We will now consider the Black approximation.

The Black approximation

The Black (1973) approximation for an American call with cash dividends is simpler than the Roll–Geske–Whaley method we have just described. For an American call option which expires at time T, with n discrete cash dividends $D_i, i = 1, \ldots, n$, at times $t_i, i = 1, \ldots, n$, it involves calculating the prices of European options that mature at times T, and t_n, and then setting the option price to the greater of these two values, see for example Hull (2003).

The Black approximation, C_{BL}, can be expressed more concisely in terms of our previously defined notation as:

$$C_{\mathrm{BL}}(S, E, \tau) = \max(v_1, v_2)$$

where v_1 and v_2 are the following European calls:

$$v_1 = c(S_D, E, \tau) \quad \text{and} \quad v_2 = c\left(S_D^+, E, \tau_1\right), \quad \tau = T - t, \ \tau_1 = T - t_n$$

and

$$S_D = S - \sum_{i=1}^{n} D_i \quad \text{and} \quad S_D^+ = S - \sum_{i=1}^{n-1} D_i$$

Code excerpt 5.2 computes the Black approximation.

Code excerpt 5.3 uses the same values as in Whaley (1981) and compares the Roll–Geske–Whaley approximation with that of Black; the results are presented in Table 5.1.

```
void black_approx(double *value, long n_divs, double dividends[], double Divs_T[],
         double S0, double X, double sigma, double T, double r, long put, long *ifail)
{
/* Input parameters:
   =================
   n_divs      - the number of dividends
   dividends[] - the dividends, dividends[0] contains the first dividend, dividend[1] the_
                 second etc.
   Divs_T[]    - the times at which the dividends are paid, Divs_T[0] is the time at which_
                 the first dividend is paid
                 Divs_T[1] is the time at which the second dividend is paid, etc.
   S0          - the current value of the underlying asset
   X           - the strike price
   sigma       - the volatility
   T           - the time to maturity
   r           - the interest rate
   put         - if put is 0 then a call option, otherwise a put option
   Output parameters:
   ==================
   value       - the value of the option, iflag - an error indicator
*/
  double zero = 0.0;
  double beta,temp,temp1,temp2,temp3;
  double tn,val_T,val_tn,tol,loc_q,err,fac;
  long i,ifailx;

  loc_q = 0.0;
  temp = 0.0;
  for (i=0; i < n_divs; ++i) {
    if (Divs_T[i] <= temp ) printf ("Error in Divs_T array, elements not increasing \n");
    if (Divs_T[i] > T) printf ("Error in Divs_T array element has a value greater than T \n");
    if (Divs_T[i] <= zero) printf ("Error in Divs_T array element <= zero \n");
    temp = Divs_T[i];
  }
  /* calculate the present value of the dividends */
  fac = 0.0;
  for (i=0; i < n_divs; ++i) {
     fac = fac + dividends[i] * exp(-r*Divs_T[i]);
  }
  temp = S0 - fac;
  /* calculate the value of the option on expiry */
  black_scholes(&val_T,NULL,temp,X,sigma,T,r,loc_q,put,&ifailx);

  /* calculate the value of the option on last dividend date */
  tn = Divs_T[n_divs-1];
  temp = temp + dividends[n_divs-1]*exp(-r*tn);
  nag_opt_bs(&val_tn,NULL,temp,X,sigma,tn,r,loc_q,putx,&ifailx);
  *value = MAX(val_tn,val_T);
}
```

Code excerpt 5.2 Function to compute the value of the Black approximation for the value of an American call option with discrete dividends.

We will now consider a more general technique for pricing both American puts and calls.

5.2.2 The MacMillan–Barone-Adesi–Whaley method

Here we consider a method of pricing American options which relies on an approximation that reduces a transformed Black–Scholes equation into a second-order ordinary differential equation, see Barone-Adesi and Whaley (1987) and MacMillan (1986). It thus provides an alternative way of evaluating American options that can be used instead of computationally intensive techniques such as finite-difference methods. Although the method prices American options, it is really based on the value of an American option *relative* to the corresponding

```
double q,r,temp,loc_r;
long i,m,m2,m_acc;
double S0,E,T,sigma,t1,delta,value,ad_value,put_value;
long is_american,ifail,put;
double bin_greeks[5],greeks[5],bin_value,bs_value;
double opt_value, critical_value, E1, E2, crit1, crit2;
double black_value;
double Divs_T[3],dividends[3];
long n_divs, put;

 E  = 100.0;
 r = 0.04;
 sigma = 0.2;
 T =  2.0;
 t1 = 1.0;
 put = 0;

/* check using the same parameters as in \inlinecite{Wha1981} */
 Divs_T[0] = 1.0;
 dividends[0] = 5.0;
 n_divs = 1;
 printf ("\nPrice S    RGW Approximation        Black Approximation  \n\n");
 for (i=0; i < 9; ++i) {
    put = 0;
    S0 = 80.0+(double)i*5.0;
    opt_RGW_approx(&opt_value,&critical_value,n_divs,dividends,Divs_T,S0,E,sigma,T,r,&ifail);
    printf("%8.4f  ",S0);
    printf("%12.3f %12.3f ",opt_value,critical_value);
    opt_black_approx(&black_value,n_divs,dividends,Divs_T,S0,E,sigma,T,r,put,&ifail);
    printf("%12.3f (%8.4e) ",black_value);
  }
```

Code excerpt 5.3 Simple test program to compare the results of function `opt_RGW_approx` with function `opt_black_approx`; the parameters used are the same as in Whaley (1981).

European option value (which can readily be computed using the Black–Scholes pricing formula).

Since an American option gives more choice, its value is always at least that of its European counterpart. This early exercise premium ($\nu(S, E, \tau) \geqslant 0$) is now defined more precisely for American puts and calls. If at current time t the asset price is S, then the early exercise premium for an American call which expires at time T, and therefore has maturity $\tau = T - t$, is:

$$\nu_c(S, E, \tau) = C(S, E, \tau) - c(S, E, \tau) \geqslant 0 \tag{5.2.8}$$

where $C(S, E, \tau)$ denotes the value of the American call and $c(S, E, \tau)$ denotes the value of the corresponding European call. The early exercise premium of an American put option, $\nu_p(S, E, \tau)$, is similarly defined as:

$$\nu_p(S, E, \tau) = P(S, E, \tau) - p(S, E, \tau) \geqslant 0 \tag{5.2.9}$$

where $P(S, E, \tau)$ is the value of the American put, and $p(S, E, \tau)$ is the value of the corresponding European put. The key insight provided by the MacMillan–Barone-Adesi–Whaley method is that, since both the American and European option values satisfy the Black–Scholes partial differential equation, so does the early exercise premium, $\nu(S, E, \tau)$; see Section 4.4.1. This means that we can write:

$$\frac{\partial \nu}{\partial t} + (r - q)S\frac{\partial \nu}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 \nu}{\partial S^2} = r\nu \tag{5.2.10}$$

where as usual S is the asset price, r is the continuously compounded interest rate, q is the continuously compounded dividend, σ is the volatility, and time t increases from the current time to the expiry time T.

We will now introduce the variable $h(\tau) = 1 - \exp(-r\tau)$ and use the factorization $v(S, E, \tau) = h(\tau)g(S, E, h)$. From standard calculus we obtain:

$$\frac{\partial v}{\partial t} = g\frac{\partial h}{\partial t} + h\frac{\partial g}{\partial t} = rg(h-1) + h\frac{\partial g}{\partial h}\frac{\partial h}{\partial t} = rg(h-1) + hr(h-1)\frac{\partial g}{\partial h}$$

and also

$$\frac{\partial v}{\partial S} = h\frac{\partial g}{\partial S} \quad \text{and} \quad \frac{\partial^2 v}{\partial S^2} = h\frac{\partial^2 g}{\partial S^2}$$

Substituting these results into Eq. (5.2.10) yields the following transformed Black–Scholes equation:

$$\frac{S^2\sigma^2 h}{2}\frac{\partial^2 g}{\partial S^2} + (r-q)Sh\frac{\partial g}{\partial S} + rg(h-1) + rh(h-1)\frac{\partial g}{\partial h} = rgh \tag{5.2.11}$$

which can be further simplified to give:

$$S^2\sigma^2\frac{\partial^2 g}{\partial S^2} + \frac{2(r-q)S}{\sigma^2}\frac{\partial g}{\partial S} - \frac{2rg}{h\sigma^2} - \frac{2r(1-h)}{\sigma^2}\frac{\partial g}{\partial h} = rgh \tag{5.2.12}$$

or

$$S^2\frac{\partial^2 g}{\partial S^2} + \beta S\frac{\partial g}{\partial S} - \frac{\alpha}{h}g - (1-h)\alpha\frac{\partial g}{\partial h} = 0 \tag{5.2.13}$$

where $\alpha = 2r/\sigma^2$ and $\beta = 2(r-q)/\sigma^2$.

We now consider the last term of Eq. (5.2.13) and note that when τ is large, $1 - h(\tau) \sim 0$. Also when $\tau \to 0$ the option is close to maturity, and the value of both the European and American options converge; this means that $v(S, E, \tau) \sim 0$ and $\frac{\partial g}{\partial h} \sim 0$. It can thus be seen that the last term is generally quite small and the MacMillan–Barone-Adesi–Whaley approximation assumes that it can be ignored. This results in the following equation:

$$S^2\frac{\partial^2 g}{\partial S^2} + \beta S\frac{\partial g}{\partial S} - \frac{\alpha}{h}g = 0 \tag{5.2.14}$$

which is a second-order differential equation with two linearly independent solutions of the form aS^γ. They can be found by substituting $g(S, E, h) = aS^\gamma$ into Eq. (5.2.14) as follows:

$$\frac{\partial g}{\partial S} = \gamma S^{\gamma-1}, \qquad \frac{\partial^2 g}{\partial S^2} = a\gamma(\gamma-1)S^{\gamma-2} = a\gamma^2 S^{\gamma-2} - a\gamma S^{\gamma-2}$$

so

$$S^2\frac{\partial^2 g}{\partial S^2} = a\gamma^2 S^\gamma - a\gamma S^\gamma = \gamma^2 g - \gamma g$$

and

$$\beta S\frac{\partial g}{\partial S} = \beta S a\gamma S^{\gamma-1} = \beta\gamma S^\gamma = \beta\gamma g$$

When the above results are substituted in Eq. (5.2.14) we obtain the quadratic equation:

$$\gamma^2 g - \gamma g + \beta\gamma g - \frac{\alpha}{h} = g\left(\gamma^2 - \gamma + (\beta - 1)\gamma - \frac{\alpha}{h}\right) = 0$$

or

$$\gamma^2 - \gamma + (\beta - 1)\gamma - \frac{\alpha}{h} = 0 \tag{5.2.15}$$

which has the two solutions

$$\gamma_1 = \frac{1}{2}\left\{-(\beta - 1) - \sqrt{(\beta - 1)^2 + 4\frac{\alpha}{h}}\right\} \tag{5.2.16}$$

and

$$\gamma_2 = \frac{1}{2}\left\{-(\beta - 1) + \sqrt{(\beta - 1)^2 + 4\frac{\alpha}{h}}\right\} \tag{5.2.17}$$

where we note that since $\alpha/h > 0$, we have $\gamma_1 < 0$ and $\gamma_2 > 0$.

The general solution to Eq. (5.2.14) is thus:

$$g(S, E, h) = a_1 S^{\gamma_1} + a_2 S^{\gamma_2} \tag{5.2.18}$$

We will now derive the appropriate solutions pertaining to American call options and American put options.

American call options

Here we use the fact that both the value and the early exercise premium ($v_c(S, E, \tau) = hg_c(S, E, h)$) of an American call tend to zero as the asset price $S \to 0$. This means that as $S \to 0$, $g_c(S, E, h) \to 0$.

However, since $\gamma_1 < 0$, the only way this can be achieved in Eq. (5.2.18) is if $a_1 = 0$. So $g_c(S, E, h) = a_2 S^{\gamma_2}$, and the value of an American call is:

$$C(S, E, \tau) = c(S, E, \tau) + ha_2 S^{\gamma_2} \tag{5.2.19}$$

An expression for a_2 can be found by considering the critical asset price (point on the early exercise boundary), S^*, above which the American option will be exercised. For $S < S^*$, the value of the American call is governed by Eq. (5.2.19), and when $S > S^*$ we have $C(S, E, \tau) = S - E$.

Now, since the value of the American option is continuous, at the critical asset value S^* the following equation applies:

$$S^* - E = c\left(S^*, E, \tau\right) + ha_2 S^{*\gamma_2} \tag{5.2.20}$$

Furthermore, since the gradient of the American option value is also continuous, at S^* we have:

$$\frac{\partial(S^* - E)}{\partial S^*} = \frac{\partial}{\partial S^*}\left\{c\left(S^*, E, \tau\right) + ha_2 S^{*\gamma_2}\right\} \tag{5.2.21}$$

which gives:

$$1 = \exp(-q\tau)N_1\left(d_1\left(S^*\right)\right) + \gamma_2 ha_2 S^{*(\gamma_2 - 1)} \tag{5.2.22}$$

where we have used the value of the hedge parameter Δ_c for a European call (see the section on the Greeks):

$$\Delta_c = \frac{\partial c(S^*, E, \tau)}{\partial S^*} = \exp(-q\tau)N_1\big(d_1\big(S^*\big)\big)$$

Equation (5.2.22) can therefore be written as:

$$ha_2 S^{*\gamma_2} = \frac{S^*}{\gamma_2}\big\{1 - \exp(-q\tau)N_1\big(d_1\big(S^*\big)\big)\big\} \tag{5.2.23}$$

When the left-hand side of the above equation is substituted into Eq. (5.2.20) we obtain the following equation for S^*:

$$S^* - E = c\big(S^*, E, \tau\big) + \frac{S^*}{\gamma_2}\big\{1 - \exp(-q\tau)N_1\big(d_1\big(S^*\big)\big)\big\} \tag{5.2.24}$$

This equation can be solved for S^* using standard iterative methods (see the section on the numerical solution of critical asset values). Once S^* has been found Eq. (5.2.23) gives:

$$ha_2 = A_2 S^{*-\gamma_2}$$

where

$$A_2 = \frac{S^*}{\gamma_2}\big\{1 - \exp(-q\tau)N_1\big(d_1\big(S^*\big)\big)\big\}$$

From Eq. (5.2.19) the value of an American call is thus of the form:

$$C(S, E, \tau) = c(S, E, \tau) + A_2\left(\frac{S}{S^*}\right)^{\gamma_2} \quad \text{when } S < S^* \tag{5.2.25}$$

$$C(S, E, \tau) = S - E \quad \text{when } S \geqslant S^* \tag{5.2.26}$$

American put options

For an American put option we proceed in a similar manner to that for the American call. We now use the fact that both the value and early exercise premium ($v_p(S, E, \tau) = hg_p(S, E, h)$) of an American put tend to zero as the asset price $S \to \infty$. So $g_p(S, E, h) \to 0$ as $S \to \infty$. Since $\gamma_2 > 0$ the only way this can be achieved by Eq. (5.2.18) is if $a_2 = 0$. This gives $g_p(S, E, h) = a_1 S^{\gamma_1}$ and the value of an American put is:

$$P(S, E, \tau) = p(S, E, \tau) + ha_1 S^{\gamma_1} \tag{5.2.27}$$

An expression for a_1 can be found by considering the critical asset price, S^{**}, below which the American option will be exercised. For $S > S^{**}$ the value of the American put is given by Eq. (5.2.27), and for $S < S^{**}$ we have $P(S, E, \tau) = E - S$.

Continuity of the American option value at the critical asset price gives:

$$E - S^{**} = p\big(S^{**}, E, \tau\big) + ha_1 S^{**\gamma_1} \tag{5.2.28}$$

and continuity of the option value's gradient at the critical asset price yields:

$$\frac{\partial(E - S^{**})}{\partial S^{**}} = \frac{\partial}{\partial S^{**}}\left\{p\left(S^{**}, E, \tau\right) + ha_1 S^{**\gamma_1}\right\} \tag{5.2.29}$$

which can be simplified to:

$$-1 = -N_1\left(-d_1\left(S^{**}\right)\right)\exp(-q\tau) + \gamma_1 a_1 S^{**(\gamma_1 - 1)} \tag{5.2.30}$$

where we have used the value of hedge parameter Δ_p for a European put (see Appendix A.3):

$$\begin{aligned}\Delta_p &= \frac{\partial p(S^{**}, E, \tau)}{\partial S^{**}} \\ &= \left\{N_1\left(d_1\left(S^{**}\right)\right) - 1\right\}\exp(-q\tau) = -N_1\left(-d_1\left(S^{**}\right)\right)\exp(-q\tau)\end{aligned}$$

Equation (5.2.30) can therefore be written as:

$$ha_1 S^{**\gamma_1} = -\frac{S^{**}}{\gamma_1}\left\{1 - N_1\left(-d_1\left(S^{**}\right)\right)\exp(-q\tau)\right\} \tag{5.2.31}$$

When the left-hand side of the above equation is substituted into Eq. (5.2.28) we obtain the following equation for S^{**}:

$$E - S^{**} = p\left(S^{**}, E, \tau\right) + \left\{1 - \exp(-q\tau)N\left[-d_1\left(S^{**}\right)\right]\right\}\frac{S^{**}}{\gamma_1} \tag{5.2.32}$$

which can be solved iteratively to yield S^{**} (see the section on the numerical solution of critical asset values). Once S^{**} has been found Eq. (5.2.31) gives:

$$ha_1 = A_1 S^{**-\gamma_1}$$

where

$$A_1 = -\left(\frac{S^{**}}{\gamma_1}\right)\left\{1 - \exp(-q\tau)N_1\left(-d_1\left(S^{**}\right)\right)\right\}$$

We note here that $A_1 > 0$ since, $\gamma_1 < 0$, $S^{**} > 0$, and $N_1(-d_1(S^{**}))\exp(-q\tau) < 1$.

From Eq. (5.2.27) the value of an American put is thus:

$$P(S, E, \tau) = p(S, E, \tau) + A_1\left(\frac{S}{S^{**}}\right)^{\gamma_2} \quad \text{when } S > S^{**}$$

$$P(S, E, \tau) = E - S \quad \text{when } S \leqslant S^{**}$$

5.2.3 Numerical solution of critical asset values

We now provide details on how to iteratively solve for the critical asset price in Eqs. (5.2.24) and (5.2.32).

American call options

For American call options we need to solve Eq. (5.2.24), which is:

$$S^* - E = c(S^*, E, \tau) + \frac{S^*}{\gamma_2}\{1 - \exp(-q\tau)N_1(d_1(S^*))\}$$

We denote the ith approximation to the critical asset value S^* by S_i^*, and represent the left-hand side of the equation by:

$$LHS(S_i^*, E, \tau) = S_i^* - E$$

and the right-hand side of the equation by:

$$RHS(S_i^*, E, \tau) = c(S_i^*, E, \tau) + \frac{S_i^*}{\gamma_2}\{1 - \exp(-q\tau)N_1(d_1(S_i^*))\}$$

If we let $K(S_i^*, E, \tau) = RHS(S_i^*, E, \tau) - LHS(S_i^*, E, \tau)$ then we want to find the value of S_i^* which (to a specified tolerance) gives $K(S_i^*, E, \tau) \sim 0$. This can be achieved with Newton's root finding method, in which a better approximation, S_{i+1}^*, can be found using:

$$S_{i+1}^* = S_i^* - \frac{K(S_i^*, E, \tau)}{K'(S_i^*, E, \tau)} \tag{5.2.33}$$

where:

$$\begin{aligned} K'(S_i^*, E, \tau) &= \frac{\partial}{\partial S_i^*}\{RHS(S_i^*, E, \tau) - LHS(S_i^*, E, \tau)\} \\ &= \frac{\partial}{\partial S_i^*}\{RHS(S_i^*, E, \tau)\} - \frac{\partial}{\partial S_i^*}\{LHS(S_i^*, E, \tau)\} \\ &= b_i - 1 \end{aligned}$$

Here we have used $b_i = \frac{\partial}{\partial S_i^*}\{RHS(S_i^*, E, \tau)\}$, and the expression for b_i is given by Eq. (5.2.35), which is derived at the end of this section.

Substituting for $K(S_i^*, E, \tau)$ and $K'(S_i^*, E, \tau)$ into Eq. (5.2.32), we therefore obtain:

$$\begin{aligned} S_{i+1}^* &= S_i^* - \frac{RHS(S_i^*, E, \tau) - LHS(S_i^*, E, \tau)}{b_i - 1} \\ &= S_i^* - \frac{RHS(S_i^*, E, \tau) - (S_i^* - E)}{b_i - 1} \\ &= \frac{b_i S_i^* - RHS(S_i^*, E, \tau) - E}{b_i - 1} \end{aligned}$$

The final iterative algorithm for the American call is therefore:

$$S_{i+1}^* = \frac{E + RHS(S_i^*, E, \tau) - b_i S_i^*}{1 - b_i} \tag{5.2.34}$$

where we can use $S_0^* = E$ for the initial estimate of the critical value (see the computer Code excerpt 5.4).

The expression for b_i

Here we derive an expression for the term b_i which is used in Eq. (5.2.34).

$$b_i = \frac{\partial c(S_i^*, E, \tau)}{\partial S_i^*} + \frac{1}{\gamma_2}\{1 - \exp(-q\tau)N_1(d_1(S_i^*))\} - \frac{S_i^*}{\gamma_2}\frac{\partial N_1(d_1(S_i^*))}{\partial d_1(S_i^*)}\frac{\partial d_1(S_i^*)}{\partial S_i^*}$$

We will now quote the following results which are derived in Appendix A:

Appendix A, Eq. (A.1.3)

$$\frac{\partial N_1(d_1(S_i^*))}{\partial d_1(S_i^*)} = n(d_1(S_i^*))$$

Appendix A, Eq. (A.1.6)

$$\frac{\partial d_1(S_i^*)}{\partial S_i^*} = \frac{1}{S_i^*\sigma\sqrt{\tau}}$$

Appendix A, Eq. (A.3.2)

$$\Delta_c = \frac{\partial c(S_i^*, E, \tau)}{\partial S_i^*} = \exp(-q\tau)N_1(d_1(S_i^*))$$

Substituting these results into the above expression, we therefore obtain:

$$b_i = \exp(-q\tau)N_1(d_1(S_i^*)) + \frac{1}{\gamma_2} - \frac{\exp(-q\tau)N_1(d_1(S_i^*))}{\gamma_2} - \frac{\exp(-q\tau)n(d_1(S_i^*))}{\gamma_2\sigma\sqrt{\tau}}$$

which can be rearranged to yield:

$$b_i = \exp(-q\tau)N_1(d_1(S_i^*))\left\{1 - \frac{1}{\gamma_2}\right\} + \frac{1}{\gamma_2}\left\{1 - \frac{\exp(-q\tau)n(d_1(S_i^*))}{\sigma\sqrt{\tau}}\right\} \tag{5.2.35}$$

American put options

For American put options we need to solve Eq. (5.2.32) which is:

$$E - S_i^{**} = p(S_i^{**}, E, \tau) - \frac{S_i^{**}}{\gamma_1}\{1 - N_1(-d_1(S_i^{**}))\exp(-q\tau)\}$$

If we let S_i^{**} denote the ith approximation to the critical asset value S^{**}, then we can represent the left-hand side of the equation by:

$$LHS(S_i^{**}, E, \tau) = E - S_i^{**}$$

and the right-hand side of the equation by:

$$\begin{aligned} RHS(S_i^{**}, E, \tau) &= p(S_i^{**}, \tau) - \frac{S_i^{**}}{\gamma_1}\{1 - N_1(-d_1(S_i^{**}))\exp(-q\tau)\} \\ &= p(S_i^{**}, E, \tau) - \frac{S_i^{**}}{\gamma_1}\{1 - [1 - N_1(d_1(S_i^{**}))]\exp(-q\tau)\} \\ &= p(S_i^{**}, E, \tau) - \frac{S_i^{**}}{\gamma_1}\{1 - \exp(-q\tau) \\ &\qquad + N_1(d_1(S_i^{**}))\exp(-q\tau)\} \end{aligned}$$

We then denote $K(S_i^{**}, E, \tau) = RHS(S_i^{**}, E, \tau) - LHS(S_i^{**}, E, \tau)$ and using Newton's method we obtain:

$$S_{i+1}^{**} = S_i^{**} - \frac{K(S_i^{**}, E, \tau)}{K'(S_i^{**}, E, \tau)} \tag{5.2.36}$$

where as before:

$$K'(S_i^{**}, E, \tau) = \frac{\partial}{\partial S_i^{**}}\{RHS(S_i^{**}, E, \tau) - LHS(S_i^{**}, E, \tau)\}$$

So $K'(S_i^{**}, E, \tau) = 1 + b_i$, where $b_i = \frac{\partial(RHS(S_i^{**}, E, \tau))}{\partial S_i^{**}}$, and the expression for b_i is given by Eq. (5.2.38), which is derived at the end of this section.

Equation (5.2.36) can therefore be written as:

$$\begin{aligned} S_{i+1}^{**} &= S_i^{**} - \frac{RHS(S_i^{**}, E, \tau) - LHS(S_i^{**}, E, \tau)}{1 + b_i} \\ &= \frac{S_i^{**}(1 + b_i) - RHS(S_i^{**}, E, \tau) + E - S_i^{**}}{1 + b_i} \end{aligned}$$

The final iterative algorithm for the American put is therefore:

$$S_i^{**} = \frac{E - RHS(S_i^{**}, E, \tau) + b_i S_i^{**}}{1 + b_i} \tag{5.2.37}$$

where we can use $S_0^{**} = E$ for the initial estimate of the critical asset value (see the computer Code excerpt 5.4).

The expression for b_i

Here we derive an expression for the term b_i which is used in Eq. (5.2.37). Since

$$b_i = \frac{\partial}{\partial S_i^{**}}\left\{p(S_i^{**}, E, \tau) - \frac{S_i^{**}}{\gamma_1}(1 - \exp(-q\tau) + N_1(d_1(S_i^{**}))\exp(-q\tau))\right\}$$

we have

$$\begin{aligned} b_i = &\frac{\partial p(S_i^{**}, E, \tau)}{\partial S_i^{**}} - \frac{1}{\gamma_1}\{1 - \exp(-q\tau)\} - \frac{1}{\gamma_1}\exp(-q\tau)N_1(d_1(S_i^{**})) \\ &- \frac{S_i^{**}\exp(-q\tau)}{\gamma_1}\frac{\partial N_1(d_1(S_i^{**}))}{\partial d_1(S_i^{**})}\frac{\partial d_1(S_i^{**})}{\partial S_i^{**}} \end{aligned}$$

We will now quote the following results which are derived in Appendix A:

Appendix A, Eq. (A.1.3):

$$\frac{\partial N_1(d_1(S_i^{**}))}{\partial d_1(S_i^{**})} = n\big(d_1\big(S_i^{**}\big)\big)$$

Appendix A, Eq. (A.1.6):

$$\frac{\partial d_1(S_i^{**})}{\partial S_i^{**}} = \frac{1}{S_i^{**}\sigma\sqrt{\tau}}$$

Appendix A, Eq. (A.3.4):

$$\Delta_p = \frac{\partial p(S_i^{**}, E, \tau)}{\partial S_i^{**}} = \exp(-q\tau)\big\{N_1\big(d_1\big(S_i^{**}\big)\big) - 1\big\}$$

Substituting these results into the above expression, we therefore obtain:

$$\begin{aligned} b_i &= \exp(-q\tau)\big\{N_1\big(d_1\big(S_i^{**}\big)\big) - 1\big\} \\ &\quad - \frac{1}{\gamma_1}\big\{1 - \exp(-q\tau) + N_1\big(d_1\big(S_i^{**}\big)\big)\exp(-q\tau)\big\} \\ &\quad - \frac{S_i^{**}\exp(-q\tau)}{\gamma_1}\frac{\partial N_1(d_1(S_i^{**}))}{\partial d_1(S_i^{**})}\frac{\partial d_1(S_i^{**})}{\partial S_i^{**}} \\ &= \exp(-q\tau)\big\{N_1\big(d_1\big(S_i^{**}\big)\big) - 1\big\} \\ &\quad - \frac{1}{\gamma_1}\big\{1 - \exp(-q\tau) + N_1\big(d_1\big(S_i^{**}\big)\big)\exp(-q\tau)\big\} \\ &\quad - \frac{S_i^{**}\exp(-q\tau)n(d_1(S_i^{**}))}{\gamma_1\sigma\sqrt{\tau}} \end{aligned}$$

which can be rearranged to yield:

$$\begin{aligned} b_i &= \exp(-q\tau)N_1\big(d_1\big(S_i^{**}\big)\big)\left\{1 - \frac{1}{\gamma_1}\right\} \\ &\quad + \frac{1}{\gamma_1}\left\{\exp(-q\tau) - 1 - \frac{\exp(-q\tau)n(d_1(S_i^{**}))}{\sigma\sqrt{\tau}}\right\} - \exp(-q\tau) \end{aligned} \tag{5.2.38}$$

In Code excerpt 5.4 we provide computer code to implement the MacMillan–Barone-Adesi–Whaley method.

```
void MBW_approx(double *opt_value, double *critical_value, double S0, double X,
                      double sigma, double T, double r, double q, long put, long *iflag)
{
/* Input parameters:
   =================
   S0              - the current value of the underlying asset
   X               - the strike price
   sigma           - the volatility
   T               - the time to maturity
   r               - the interest rate
```

Code excerpt 5.4.

```
   q               - the continuous dividend yield
   put             - if put is 0 then a call option, otherwise a put option
   Output parameters:
   ==================
   opt_value       - the value of the option
   critical_value  - the critical value
   iflag           - an error indicator
*/
  double A_1,A_2,S_star,gamma_2,gamma_1;
  double d1,alpha,h,beta,temp,temp1;
  double pdf,pi,b,rhs,eur_val,tol,err;
  long iterate;
  long iflagx,putx;

  pi = PI;
  beta = 2.0 * (r - q) / (sigma * sigma);
  alpha = 2.0 * r / (sigma * sigma);
  h = 1.0 - exp(-r*T);
  temp = beta - 1.0;
  iterate = 1;
  tol = 0.000001;
  if (!put) {  /* An American call */
    gamma_2 = (-temp + sqrt((temp*temp) + (4.0*alpha/h)));
    gamma_2 = gamma_2 / 2.0;
    S_star = X;
    while (iterate) {  /* calculate S_star, iteratively */
      d1 = log(S_star/X) + (r-q+(sigma*sigma/2.0))*T;
      d1 = d1/(sigma*sqrt(T));
      pdf = (1.0/sqrt(2.0*pi))*exp(-d1*d1/2.0);
      temp = exp (-q*T)*cum_norm(d1)*(1.0 - (1.0/gamma_2));
      temp1 = (1.0 - ((exp(-q*T)*pdf)/(sigma*sqrt(T))))/gamma_2;
      b = temp + temp1;
      /* calculate the Black-Scholes value of a European call */
      putx = 0;
      black_scholes(&eur_val,NULL,S_star,X,sigma,T,r,q,putx,&iflagx);
      rhs = eur_val+(1.0-exp(-q*T)*cum_norm(d1))*S_star/gamma_2;
      S_star = (X + rhs - b*S_star)/(1.0-b);
      err = fabs((S_star - X) - rhs)/X;
        if (err < tol) iterate = 0;
    }
    A_2 = (S_star/gamma_2)*(1.0 - exp(-q*T)*cum_norm(d1));
    if (S0 < S_star) {
       temp1 = S0/S_star;
       black_scholes(&temp,NULL,S0,X,sigma,T,r,q,putx,&iflagx);
       *opt_value = temp + A_2 * pow(temp1,gamma_2);
    }
    else {
       *opt_value = S0 - X;
    }
  }
  else { /* An American put */
    gamma_1 = (-temp - sqrt((temp*temp) + (4.0*alpha/h)));
    gamma_1 = gamma_1 / 2.0;
    S_star = X;
    while (iterate) {  /* calculate S_star, iteratively */
      d1 = log(S_star/X) + (r-q+(sigma*sigma/2.0))*T;
      d1 = d1/(sigma*sqrt(T));
      pdf = (1.0/sqrt(2.0*pi))*exp(-d1*d1/2.0);
      temp = exp(-q*T)*(cum_norm(d1)*(1.0-(1.0/gamma_1))-1.0);
      temp1 = (exp(-q*T)-1.0-((exp(-q*T)*pdf)/(sigma*sqrt(T))))/gamma_1;
      b = temp + temp1;
      /* calculate the Black-Scholes value of a European put */
      putx = 1;
      black_scholes(&eur_val,NULL,S_star,X,sigma,T,r,q,putx,&iflagx);
      rhs = eur_val-(1.0-exp(-q*T)+exp(-q*T)*cum_norm(d1))*S_star/gamma_1;
      S_star = (X - rhs + b*S_star)/(1.0+b);
      err = fabs((X - S_star) - rhs)/X;
      if (err < tol) iterate = FALSE;
    }
    A_1 = -(S_star/gamma_1)*(1.0 - exp(-q*T)*cum_norm(-d1));
    if (S0 > S_star) {
       temp1 = S0/S_star;
       black_scholes(&temp,NULL,S0,X,sigma,T,r,q,putx,&iflagx);
       *opt_value = temp + A_1 * pow(temp1,gamma_1);
    }
```

Code excerpt 5.4 (*Continued*).

```
    else {
       *opt_value = X - S0;
    }
  }
  *critical_value = S_star;
}
```

Code excerpt 5.4 Function to compute the MacMillan–Barone-Adesi–Whaley approximation for American options.

Table 5.2 The MacMillan–Barone-Adesi–Whaley method for American option values computed by the routine `MBW_approx`

Stock price	Call		Put	
	Accurate value	Error	Accurate value	Error
86.0	1.2064	5.54×10^{-4}	14.0987	-3.69×10^{-2}
89.0	1.8838	1.95×10^{-4}	11.5120	-4.85×10^{-2}
92.0	2.7890	7.03×10^{-4}	9.2478	-3.58×10^{-2}
95.0	3.9427	1.16×10^{-3}	7.3031	-1.66×10^{-2}
98.0	5.3522	1.15×10^{-3}	5.6674	7.19×10^{-4}
101.0	7.0119	1.10×10^{-3}	4.3209	1.35×10^{-2}
104.0	8.9043	2.21×10^{-3}	3.2362	2.22×10^{-2}
107.0	11.0072	2.63×10^{-3}	2.3823	2.63×10^{-2}
110.0	13.2905	4.20×10^{-3}	1.7235	2.80×10^{-2}
113.0	15.7264	4.77×10^{-3}	1.2272	2.66×10^{-2}

The parameters used were: $\tau = 0.5$, $X = 100.0$, $r = 0.1$, $q = 0.06$, $\sigma = 0.2$. The accurate value was calculated using a standard lattice with 2000 time steps, and the error was the MacMillan–Barone-Adesi–Whaley estimate minus the accurate value.

Table 5.3 The MacMillan, Barone-Adesi, and Whaley critical asset values for the early exercise boundary of an American put computed by the routine `MBW_approx`

Time to expiry, τ	Critical asset value, S^{**}	Time to expiry, τ	Critical asset value, S^{**}
1.00	82.1510	0.50	85.1701
0.95	82.3751	0.45	85.6199
0.90	82.6115	0.40	86.1176
0.85	82.8618	0.35	86.6740
0.80	83.1273	0.30	87.3049
0.75	83.4098	0.25	88.0333
0.70	83.7115	0.20	88.8959
0.65	84.0349	0.15	89.9568
0.60	84.3830	0.10	91.3469
0.55	84.7598	0.05	93.4260

The parameters used were: $S = 101.0$, $X = 101.0$, $r = 0.1$, $q = 0.06$, and $\sigma = 0.20$.

The results given in Tables 5.2 and 5.3 were obtained by using the function `MBW_approx`.

5.3 Lattice methods for vanilla options

5.3.1 Binomial lattice

In this section we will derive equations for a binomial lattice that describes the GBM movement of asset price changes. The approach that we will adopt is based on the work of Cox, Ross, and Rubinstein (1979) and will be referred to as the CRR lattice.

From Chapter 2, Eq. (2.3.9), we know that if the price of an asset, S_t, follows GBM then the change in value of its price over time interval Δt has the following distribution:

$$\log\left(\frac{S_{t+\Delta t}}{S_t}\right) \sim \mathrm{N}\left(\left(r - \frac{\sigma^2}{2}\right)\Delta t, \sigma^2 \Delta t\right)$$

If we use the notation:

$$X = \frac{S_{t+\Delta t}}{S_t}$$

and

$$\eta = \left(r - \frac{\sigma^2}{2}\right)\Delta t, \qquad \nu^2 = \sigma^2 \Delta t$$

the above equation becomes:

$$\log(X) \sim \mathrm{N}(\eta, \nu^2)$$

or equivalently

$$X \sim \Lambda(\eta, \nu^2)$$

where $\Lambda(\eta, \nu^2)$ is the lognormal distribution *derived* from a Gaussian distribution with mean η and variance ν^2. It is well known—see for example Evans, Hastings, and Peacock (2000)—that the first two moments of a variable X drawn from a lognormal distribution are:

Lognormal mean

$$E[X] = \exp\left(\eta + \frac{\nu^2}{2}\right) \tag{5.3.1}$$

substituting for η and ν^2 gives:

$$E[X] = \exp\left\{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \frac{\sigma^2}{2}\Delta t\right\} \tag{5.3.2}$$

Lognormal variance

$$\begin{aligned}\mathrm{Var}[X] &= E\big[\big(X - E[X]\big)^2\big] = E\big[X^2\big] - \big(E[X]\big)^2 \\ &= \exp\big(2\eta + \nu^2\big)\big\{\exp\big(\nu^2\big) - 1\big\}\end{aligned} \tag{5.3.3}$$

substituting for η and ν^2 gives:

$$\mathrm{Var}[X] = \exp\left\{2r\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma^2 \Delta t\right\}$$

which can be simplified to yield:

$$\mathrm{Var}[X] = \exp\{2r\Delta t\}\big\{\exp\big(\sigma^2 \Delta t\big) - 1\big\} \tag{5.3.4}$$

Since we can assume that the expected value of X grows at the riskless interest rate, r, we can also write:

$$E[X] = \exp(r\Delta t) \tag{5.3.5}$$

The above results can be used to find the first two moments of the asset price distribution $S_{t+\Delta t}$, given that we know the asset price, S_t, at time instant t. To do this we will use (see Appendix C.3 for a proof) the fact that for a random variable G we have:

$$E[a + bG] = E[a] + bE[G] \quad \text{and} \quad \mathrm{Var}[a + bG] = b^2\,\mathrm{Var}[G]$$

where a and b are constants. Applying this to the variable X gives:

$$E[X] = E\left[\frac{S_{t+\Delta t}}{S_t}\right] = \frac{1}{S_t} E[S_{t+\Delta t}] \tag{5.3.6}$$

and

$$\mathrm{Var}[X] = \mathrm{Var}\left[\frac{S_{t+\Delta t}}{S_t}\right] = \frac{1}{S_t^2}\,\mathrm{Var}[S_{t+\Delta t}] \tag{5.3.7}$$

where we have used $a = 0$ and $b = \dfrac{1}{S_t}$. Note: It is also easy to show that:

$$\mathrm{Var}[S_{t+\Delta t}] = \mathrm{Var}[\Delta S] \tag{5.3.8}$$

where the change in asset price over the time interval Δt is denoted by $\Delta S = S_{t+\Delta t} - S_t$. This elementary result sometimes is used without proof, see for example Hull (1997), p. 344. The proof is simple:

$$\mathrm{Var}[S_{t+\Delta t}] = \mathrm{Var}[S_t + \Delta S] = \mathrm{Var}[\Delta S]$$

where again we have used:

$$\mathrm{Var}[a + bG] = b^2\,\mathrm{Var}[G], \quad \text{this time with } a = 0 \text{ and } b = 1.$$

To find expressions for the mean and variance of $S_{t+\Delta t}$ we simply substitute Eq. (5.3.5) into Eq. (5.3.6) and obtain:

$$E[S_{t+\Delta t}] = S_t \exp(r\Delta t) \tag{5.3.9}$$

and substituting Eq. (5.3.4) into Eq. (5.3.7) gives:

$$\text{Var}[S_{t+\Delta t}] = S_t^2 \exp(2r\Delta t)\{\exp(\sigma^2\Delta t) - 1\} \tag{5.3.10}$$

Since we are modelling asset price movements with a binomial lattice, the asset price, S_t, at any given node is only permitted to either *jump up* or *jump down* in value over the next time step Δt. Here we will assume that the new asset price, $S_{t+\Delta t}$, is $S_t u$ for an up jump and $S_t d$ for a down jump where u and d are constants that apply to all lattice nodes. If we further denote the probability of an up jump by p, then the probability of a down jump must (by definition) be $1 - p$.

Now that we have specified the lattice parameters we will use these to match the first two moments of the lognormal distribution. This results in the following equation for the mean:

$$E[S_{t+\Delta t}] = pS_t u + (1-p)S_t d = S_t \exp(r\Delta t) \tag{5.3.11}$$

The corresponding equation for the variance requires a little more work:

$$\text{Var}[S_{t+\Delta t}] = E\left[(S_{t+\Delta t})^2\right] - \left(E[S_{t+\Delta t}]\right)^2 \tag{5.3.12}$$

Since

$$E\left[(S_{t+\Delta t})^2\right] = p(S_t u)^2 + (1-p)(S_t d)^2 = S_t^2\left\{pu^2 + (1-p)d^2\right\} \tag{5.3.13}$$

and, from Eq. (5.3.9), we have:

$$\left(E[S_{t+\Delta t}]\right)^2 = \left\{S_t \exp(r\Delta t)\right\}^2 = S_t^2 \exp(2r\Delta t) \tag{5.3.14}$$

we can substitute Eqs. (5.3.13) and (5.3.14) into Eq. (5.3.12) to obtain:

$$\text{Var}[S_{t+\Delta t}] = S_t^2\left\{pu^2 + (1-p)d^2\right\} - S_t^2 \exp(2r\Delta t) \tag{5.3.15}$$

So from Eqs. (5.3.10) and (5.3.15):

$$\begin{aligned}\exp(2r\Delta t)\{\exp(\sigma^2\Delta t) - 1\} &= pu^2 + (1-p)d^2 \\ &= pu^2 + (1-p)d^2 - \exp(2r\Delta t)\end{aligned} \tag{5.3.16}$$

So, restating Eq. (5.3.11) and simplifying Eq. (5.3.16), we obtain the following two equations:

$$pu + (1-p)d = \exp(r\Delta t) \tag{5.3.17}$$

$$\exp(2r\Delta t + \sigma^2\Delta t) = pu^2 + (1-p)d^2 \tag{5.3.18}$$

which we will use to solve for the three parameters u, d, and p. Since there are three unknowns and only two equations, we can impose an additional constraint to obtain a unique solution. The constraint used in the CRR binomial model is:

$$u = \frac{1}{d}$$

We now use the following notation:

$$a = \exp(r\Delta t)$$

and

$$b^2 = \exp(2r\Delta t)\{\exp(\sigma^2\Delta t) - 1\} = a^2\{\exp(\sigma^2\Delta t) - 1\}$$

This means that Eq. (5.3.17) can be written as:

$$a = pu + (1 - p)d$$

which gives:

$$p = \frac{a - d}{u - d} \tag{5.3.19}$$

From Eq. (5.3.18) we have:

$$\exp(2r\Delta t + \sigma^2\Delta t) = a^2\exp(\sigma^2\Delta t) = a^2 + b^2$$

and so:

$$a^2 + b^2 = pu^2 + (1 - p)d^2$$

Rearranging we have:

$$pu^2 + (1 - p)d^2 - a^2 = b^2$$

$$pu^3 + (1 - p)d^2u - a^2u - b^2u = 0$$

but:

$$(1 - p)d^2u = (1 - p)d = a - pu$$

so

$$pu^3 + (a - pu) - a^2u - b^2u = 0$$

or

$$p(u^3 - u) + a - a^2u - b^2u = 0$$

Now,

$$p(u^3 - u) = u^2p(u - d) = u^2(a - d) = u^2a - u$$

which gives:

$$au^2 - u + a - a^2u - b^2u = 0$$

So we obtain the following quadratic equation in u:

$$au^2 - u(1 + a^2 + b^2) + a = 0$$

The solution is:

$$u = \frac{(1 + a^2 + b^2) + \sqrt{(1 + a^2 + b^2)^2 - 4a^2}}{2a}$$

If Δt is small we can obtain a *reasonable approximation* to the solution by neglecting terms of order higher than Δt.

In these circumstances we have:

$$a^2 + b^2 + 1 = \exp(2r\Delta t) + \exp(2r\Delta t)\left\{\exp\left(\sigma^2\Delta t\right) - 1\right\} + 1$$
$$\sim 1 + 2r\Delta t + (1 + 2r\Delta t)\sigma^2\Delta t + 1 \sim 2 + 2r\Delta t + \sigma^2\Delta t$$

Therefore,

$$\sqrt{\left(a^2 + b^2 + 1\right)^2 - 4a^2} \sim \sqrt{\left(2 + 2r\Delta t + \sigma^2\Delta t\right)^2 - 4(1 + 2r\Delta t)}$$
$$\sim \sqrt{4 + 8r\Delta t + 4\sigma^2\Delta t - 4 - 8r\Delta t}$$
$$= \sqrt{4\sigma^2\Delta t} = 2\sigma\sqrt{\Delta t}$$

and so

$$u \sim \frac{2 + 2r\Delta t + \sigma^2\Delta t + 2\sigma\sqrt{\Delta t}}{2\exp(r\Delta t)}$$

$$u \sim \left(1 + r\Delta t + \frac{\sigma^2\Delta t}{2} + \sigma\sqrt{\Delta t}\right)(1 - r\Delta t)$$

$$u \sim 1 + r\Delta t + \frac{\sigma^2\Delta t}{2} + \sigma\sqrt{\Delta t} - r\Delta t = 1 + \sigma\sqrt{\Delta t} + \frac{\sigma^2\Delta t}{2}$$

which to order Δt gives:

$$u = \exp\left(\sigma\sqrt{\Delta t}\right) \quad \text{and} \quad d = \exp\left(-\sigma\sqrt{\Delta t}\right) \tag{5.3.20}$$

where we have used

$$\exp\left(\sigma\sqrt{\Delta t}\right) = 1 + \sigma\sqrt{\Delta t} + \frac{\sigma^2\Delta t}{2} + \frac{\sigma^3(\Delta t)^{3/2}}{6} + \cdots$$

and

$$d = \frac{1}{u}$$

It is interesting to note (by substituting into Eq. (5.3.19)) that when $r = 0$ and $\Delta t \to 0$, we have $p \to \frac{1}{2}$.

Now that we know the values of the lattice parameters u, d, and p we can use these to build a lattice with a specified number of time steps. Once this has been constructed, it can be used to compute the values and Greeks for various types of financial options. These could simply be American/European vanilla options, or more exotic options that may incorporate features such as: *lockout periods*, *barriers*, and nonstandard payoff functions.

We will now discuss how to create a lattice which can be used to value American and European vanilla options.

If the current value of the underlying asset is S, and the duration of the option is τ and we use a lattice with n equally spaced time intervals Δt, then we have:

$$\Delta t = \frac{\tau}{n}$$

The values of the asset price at various nodes in the lattice can easily be computed. This is illustrated, in Fig. 5.1, for a lattice with six time steps (that is seven lattice levels).

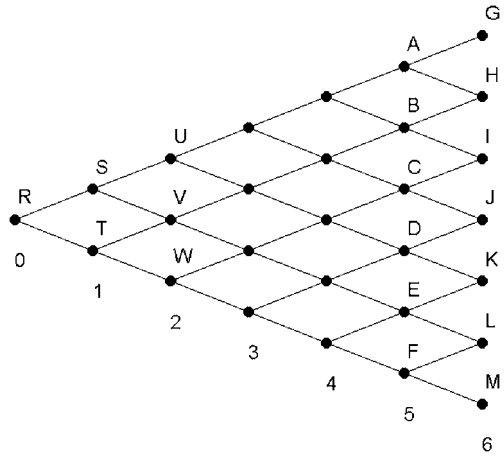

Figure 5.1 A standard binomial lattice consisting of six time steps. The root lattice node R corresponds to the current time t, and the terminal nodes G to M are those at option maturity; that is time $t + \tau$, where τ is the duration of the option. The asset value at node R is S, where S is the current asset value. Asset values at other nodes are, for example, node S: Su, node T: Sd, node V: S, and node A: Su^5. Option values are computed using a backward iterative process: the option values at nodes A–F on the penultimate time step are computed from the payouts of the terminal nodes G–M, and this process continues until the root node is reached which yields the current value of the option. Here we compute the Greeks using the following nodes: Delta uses nodes S and T, Gamma uses nodes U, V, and W, and Theta uses nodes R and V.

The asset values at the labelled nodes are:

Lattice level 1: Time t

$$S_R = S$$

Lattice level 2: Time $t + \Delta t$

$$S_S = Su, \qquad S_T = Sd$$

Lattice level 6: Time $t + 5\Delta t$

$$S_A = Su^5, \qquad S_B = Su^3, \qquad S_C = Su,$$
$$S_D = Sd, \qquad S_E = S, \qquad S_F = Sd^5$$

Lattice level 7: Time $t + 6\Delta t$

$$S_G = Su^6, \qquad S_H = Su^4, \qquad S_I = Su^2, \qquad S_J = S,$$
$$S_K = Sd^2, \qquad S_L = Sd^4, \qquad S_M = Sd^6$$

In general, at time $t + i\Delta t$, there are $i + 1$ stock prices; these are:

$$S_{i,j} = Su^j d^{i-j}, \quad j = 0, 1, \ldots, i$$

We note that, since $u = 1/d$, an up movement followed by a down movement gives the same stock price as a down movement followed by an up movement; for instance, $Su^2d = Su$. This means that the tree recombines, and the number of nodes required to represent all the different asset prices is significantly reduced.

5.3.2 Constructing and using the binomial lattice

In this section we are concerned with the practical details of how to construct, and then use, a *standard* one-dimensional binomial lattice to value American and European options. Since this lattice forms the basis for other one-dimensional and multidimensional lattice techniques, we will discuss its construction in some detail. A complete computer program for a standard binomial lattice is given in Code excerpt 5.11, and we will use this as a basis for our discussions. The results of using this code are presented in Fig. 5.2. For

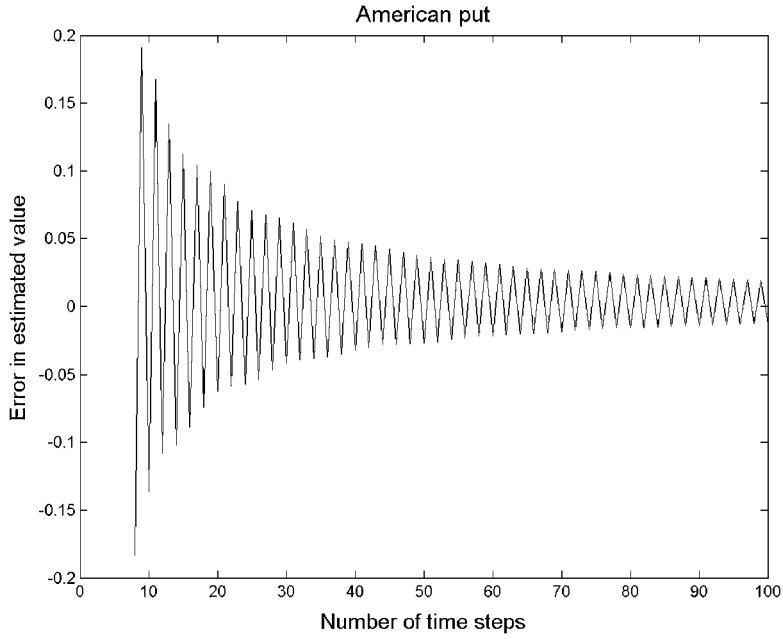

Figure 5.2 The error in the estimated value, *est_val*, of an American put using a standard binomial lattice. The parameters used were: $T = 1.0$, $S = 105.0$, $X = 105.0$, $r = 0.1$, $q = 0.02$, $\sigma = 0.3$. The very accurate value (*acc_val*) was 9.2508 and was computed using a 6000 step standard binomial lattice. The error in the estimated value was obtained as *est_val* − *acc_val*.

easy reference we will now list the input parameters used by this computer program:

S0	the current price of the underlying asset, S
X	the strike price
sigma	the volatility of the asset
T	the maturity of the option in years
r	the risk-free interest rate
q	the continuous dividend yield
put	if put equals 1 then the option is a put option, if put equals 0 then it is a call option
is_american	if is_american equals 1 then it is an American option, if is_american equals 0 then it is a European option
M	the number of time steps in the lattice

We will now discuss in more detail the computational issues involved in each stage of the calculation.

Compute the values of the constants used by the lattice

First calculate the values of various constants that will be used, see Code excerpt 5.5.

For convenience, we have used the variables p_u and p_d to store, respectively, the up and down jump probabilities discounted by the interest rate r over one time step; these values will be used later on when we work backwards through the lattice to calculate the current option value.

Assign the asset values to the lattice nodes

We will now show that the number of different asset prices, $\mathcal{LS}_n$, for an n step recombining lattice is $2n + 1$.

The nodes in a recombining lattice can be considered as being composed of two kinds: those corresponding to an *even* time step, and those corresponding to an *odd* time step.

This is because the set of node asset values, $\mathcal{ET}$, for an even time step is distinct from the set of node asset values, $\mathcal{OT}$, for an odd time step. Although

```
dt = T/(double)M;
t1 = sigma*sqrt(dt);
u = exp(t1);
d = exp(-t1);
a = exp((r-q)*dt);
p = (a - d)/(u - d);
if ((p < zero) || (p > 1.0)) printf ("Error p out of range\n");
discount = exp(-r*dt);
p_u = discount*p;
p_d = discount*(1.0-p);
```

Code excerpt 5.5 A code fragment which computes the values of various lattice constants.

```
s[M] = S0;
for (i = 1; i <= M; ++i) {
   s[M+i] = u*s[M+i-1];
   s[M-i] = d*s[M-i+1];
}
```

Code excerpt 5.6 A code fragment which assigns the different binomial lattice asset values to the storage array `s` by using the up and down jump ratios u and d defined in Section 3.4.1. The current asset value S is assigned to the central array element `s[M]`, where `M` is the number of time steps in the lattice. The array elements above center are `S[M+i]` $= Su^i, i = 1, \ldots,$`M`, and the array elements below center are `S[M-i]` $= Sd^i, i = 1, \ldots,$`M`.

$\mathcal{ET} \cap \mathcal{OT} = \emptyset$, the elements of $\mathcal{ET}$ and $\mathcal{OT}$ for any consecutive pair of time steps are related by the simple constant multiplicative factor d. Also, for an even time step there is a central node corresponding to the current asset price `S0`, and the remaining nodes are symmetrically arranged about this, see Code excerpt 5.6. These features are illustrated in Fig. 5.1, for a standard lattice with six time steps.

The number of distinct asset prices in a lattice is therefore the sum of the number of nodes in the last two time steps. Since the number of nodes in the ith time step, $\mathcal{S}_i$, is $i + 1$ (see Fig. 5.1), for an n time step lattice we have:

$$\mathcal{S}_n = n + 1 \quad \text{and} \quad \mathcal{S}_{n-1} = n$$

This means that the number of different asset values in an n time step lattice is:

$$\mathcal{LS}_n = \mathcal{S}_n + \mathcal{S}_{n-1} = 2n + 1$$

The number of nodes in an n time step lattice, $\mathcal{LN}_n$, is:

$$\mathcal{LN}_n = \sum_{i=0}^{n} (i + 1) = \frac{(n + 1)(n + 2)}{2}$$

where we have used the fact that $\mathcal{LN}_n$ is the sum of an arithmetic progression with first term 1, increment 1 and last term $n + 1$.

One might initially think that, in order to price options, it is necessary to store the asset value of each lattice node which would entail storing $\mathcal{LN}_n$ values. However, this is not the case. We only need to store the number of *different* asset values in the lattice; that is, $\mathcal{LS}_n$ values.

Storing $\mathcal{LS}_n$ values instead of $\mathcal{LN}_n$ can result in dramatic economies of storage. For example, an accurate, 1000 step lattice, has $\mathcal{LN}_n = 2001 \times 2002 \times 1/2 = 2003001$, while the corresponding value of $\mathcal{LS}_n$ is only $2 \times 1000 + 1 = 2001$.

Compute the option payoff at the terminal nodes

The current value of an option is evaluated by starting at option maturity, the end of the tree, and working backwards. The option values for the *terminal nodes* of the tree are just given by the payoff (at maturity) of the option; this

```
if (((M+1)/2) == (M/2)) { /* then M is even */
   if (put)
     v[M/2] = MAX(X - s[M],zero);
   else
      v[M/2] = MAX(s[M]-X,zero);
}
P1 = 2*M;
P2 = 0;
for (i = 0; i < (M+1)/2; ++i) {
  if (put) {
     v[M-i] = MAX(X - s[P1],zero);
     v[i]   = MAX(X - s[P2],zero);
  }
  else {
     v[M-i] = MAX(s[P1]-X,zero);
     v[i]   = MAX(s[P2]-X,zero);
  }
  P1 = P1 - 2;
  P2 = P2 + 2;
}
```

Code excerpt 5.7 A code fragment that computes the payouts for puts and calls at the lattice terminal nodes. The payouts are assigned to elements of the array `v` and are computed using the strike price, `X`, and the previously computed asset values stored in array `s`; as before `M` is the number of time steps in the lattice.

is independent of whether the option is an American or European. For a lattice with n time steps there are $n + 1$ terminal nodes, with option values, $f_{n,j}$, $j = 0, \ldots, n$.

To compute the values of vanilla American and European options, with exercise price E, then we will start with the following terminal node values:

for put options:

$$f_{n,j} = \max\bigl(E - Su^j d^{n-j}, 0\bigr), \quad j = 0, \ldots, n,$$

and for call options:

$$f_{n,j} = \max\bigl(Su^j d^{n-j} - E, 0\bigr), \quad j = 0, \ldots, n$$

The computer code used to achieve this is Code excerpt 5.7.

Iterate backwards through the lattice

The probability of moving from node (i, j) at time $i\Delta t$ to node $(i + 1, j + 1)$ at time $(i + 1)\Delta t$ is p, and the probability of moving from node (i, j) at time $i\Delta t$ to the node $(i + 1, j)$ at time $(i + 1)\Delta t$ is $1 - p$. If we assume that there is no early exercise then:

$$f^E_{i,j} = \exp(-r\Delta t)\bigl\{pf_{i+1,j+1} + (1-p)f_{i+1,j}\bigr\}, \quad j \leqslant i \leqslant n-1,\ 0 \leqslant j \leqslant i \tag{5.3.21}$$

When early exercise, for an American option, is taken into account we have:

$$f_{i,j}^{A} = \max\left\{E - S_{i,j},\, f_{i,j}^{E}\right\} \tag{5.3.22}$$

or for an American call option:

$$f_{i,j}^{A} = \max\left\{S_{i,j} - E,\, f_{i,j}^{E}\right\}, \quad j \leqslant i \leqslant N-1,\ 0 \leqslant j \leqslant i, \tag{5.3.23}$$

where $f_{i,j}^{E}$ is given by Eq. (5.3.21).

Code excerpt 5.8 works backward through the lattice and uses the array `v` to store the option values.

At each time step the newly calculated option values overwrite those computed by the previous time step. This process is continued until the second time step (`m1 = 2`) is reached. A different technique is then used, which doesn't overwrite the option values and thus allows the Greeks to be computed in the vicinity of the root lattice node `R`. If the Greeks are not required, continue working backward through the lattice until the root node `R` (`m1 = 0`) is reached, and the current value of the option is then given by `v[0]` (or its multidimensional equivalent).

The option values at all lattice nodes in time steps 0, 1, and 2 are made accessible by the Code excerpt 5.9.

```
P2 = 0;
for (m1 = M-1; m1 >= 2; --m1) {
  P2 = P2 + 1;
  P1 = P2;
  for (n =0; n <= m1; ++n) {
    if ((v[n] == zero) && (v[n+1] == zero))  {
         hold = zero;
    }
    else
         hold = p_d*v[n] + p_u*v[n+1];
     if (is_american) {
       if (put)
         v[n] = MAX(hold,X-s[P1]);
       else
         v[n] = MAX(hold,s[P1]-X);
     }
     else
       v[n] = hold;
     P1 = P1 + 2;
   }
}
```

Code excerpt 5.8 Computer code that works iteratively backward through the lattice computing the option values at each time step. The array `v` contains the option values computed from the previous time step, and these are overwritten with option values computed for the current time step. The iteration stops at the second time step, since we do not want to overwrite values in the array `v` which are required for calculating the Greeks in the neighborhood of the root node.

```
jj = 2;
for (m1 = 2; m1 >= 1; --m1) {
  ind = M-m1+1;
  for (n =0; n < m1; ++n) {
     hold =  p_d*v[5-jj-m1-1] + p_u*v[5-jj-m1];
     if (is_american) {
       if (put)
         v[5-jj] = MAX(hold,X-s[ind]);
       else
         v[5-jj] = MAX(hold,s[ind]-X);
     }
     else
       v[5-jj] = hold;
       --jj;
       ind = ind + 2;
   }
}
*value = v[5];
```

Code excerpt 5.9 Code fragment illustrating how the option values are stored for the first two time steps so that the Greeks can be computed in the vicinity of the root node R.

Table 5.4 Lattice node values in the vicinity of the root node R

Node	Time step	Asset array element	Asset value	Option array element
R	0	s[M]	S	v[5]
S	1	s[M+1]	Su	v[4]
T	1	s[M-1]	Sd	v[3]
U	2	s[M+2]	Su^2	v[2]
V	2	s[M]	S	v[1]
W	2	s[M-2]	Sd^2	v[0]

Computing the Greeks: Δ, Γ and Θ

We will now describe how to calculate the option's hedge statistics (Greeks).

Let the option value and asset value at lattice node k be denoted by f_k and S_k respectively. So, for instance, S_T represents the asset price at node T and f_T is the corresponding option value at node T. Table 5.4 supplies details of the lattice node values in the vicinity of the root node R.

The computation of each Greek is now considered.

Delta

The definition of Δ is the rate of change of the option value with asset price all other parameters remaining fixed. Thus,

$$\Delta = \frac{\partial f}{\partial S} = \frac{\Delta f}{\Delta S}$$

where Δf is the change option value corresponding to the change in the asset price ΔS. Ideally we would like to evaluate this partial derivative at the root node R (m1=0); however, we cannot because we need at least two lattice nodes to compute a value. The best we can do is to evaluate the derivative at the first time step (m1=1) as follows:

$$\Delta = \frac{f_S - f_T}{S_S - S_T} = \frac{\text{v[4]} - \text{v[3]}}{\text{s[M} + 1\text{]} - \text{s[M} - 1\text{]}}$$

Gamma

The definition of Γ is the rate of change of Δ with asset price all other parameters remaining fixed. Thus,

$$\Gamma = \frac{\partial^2 f}{\partial S^2} = \frac{\partial \Delta}{\partial S}$$

In order to evaluate Γ we require at least two values of Δ. The nearest this can be achieved to the root node R is at time step 2, where we have:

$$\Gamma = \frac{\Delta^*_{UV} - \Delta^*_{VW}}{S^*_{UV} - S^*_{VW}}$$

with the midpoints

$$S^*_{UV} = \frac{1}{2}\{S_U + S_V\}$$

and the values of Δ at the midpoints S^*_{UV} and S^*_{VW} denoted by Δ^*_{UV} and Δ^*_{VW}, respectively. Since

$$\Delta^*_{UV} = \frac{f_U - f_V}{S_U - S_V}$$

$$\Delta^*_{VW} = \frac{f_V - f_W}{S_V - S_W}$$

and

$$S^*_{UV} - S^*_{VW} = \frac{1}{2}\{S_U - S_W\}$$

we have

$$\Delta^*_{UV} = \frac{\mathtt{v[2]} - \mathtt{v[1]}}{\mathtt{s[M+2]} - \mathtt{s[M]}}$$

$$\Delta^*_{VW} = \frac{\mathtt{v[1]} - \mathtt{v[0]}}{\mathtt{s[M]} - \mathtt{s[M-2]}}$$

The value of Γ can therefore be approximated as:

$$\Gamma = \frac{2\{\Delta^*_{UV} - \Delta^*_{VW}\}}{\mathtt{s[M+2]} - \mathtt{s[M-2]}}$$

Theta

The definition of Θ is the rate of change of option value with time all other parameters remaining fixed. Thus,

$$\Theta = \frac{\partial f}{\partial t} = \frac{\Delta f}{\Delta t}$$

The nearest to the root node R this can be computed is over the time interval from time step 0 to time step 2. We then obtain the following approximation:

$$\Theta = \frac{f_V - f_R}{2\Delta t} = \frac{\mathtt{v[1]} - \mathtt{v[5]}}{2\Delta t}$$

Code excerpt 5.10 computes the Δ, Γ, and Θ by using the approximations we have just discussed.

Vega

The definition of $\mathcal{V}$ is the rate of change of the option value with volatility:

$$\mathcal{V} = \frac{\partial f}{\partial \sigma}$$

In a standard binomial lattice $\mathcal{V}$ cannot be computed directly. A simple approach is to use two binomial lattices as follows:

$$\mathcal{V} = \frac{f_{\sigma+\Delta\sigma} - f_{\sigma}}{\Delta\sigma}$$

where $f_{\sigma+\Delta\sigma}$ is the option value computed using a binomial lattice with volatility $\sigma + \Delta\sigma$, and f_{σ} is the option value computed using another binomial lattice with a volatility of σ; all other lattice parameters remain constant.

The implied volatility of American options can be computed using the method outlined for European options in Section 5.4.4; however, in this case the option value and Greeks are computed using a binomial lattice (see Code excerpt 5.11).

```
/* assign the value of delta (obtained from m1 = 1) */
  greeks[1] = (v[4]-v[3])/(s[M+1]-s[M-1]);
 /* assign the value of gamma (use the values at time step m1 = 2) */
  dv1 = v[2] - v[1];
  ds1 = s[M+2] - s[M];
  dv2 = v[1] - v[0];
  ds2 = s[M] - s[M-2];
  h = 0.5*(s[M+2] - s[M-2]);
  greeks[0] = ((dv1/ds1) - (dv2/ds2))/h;
  /* assign the value of theta */
  greeks[2] = (v[1]-*value)/(2.0*dt); /* can also write: greeks[2] = (v[1]-v[5])/(2.0*dt); */
}
```

Code excerpt 5.10 A code fragment that computes the values of the Greeks, Delta, Gamma and Theta, in the vicinity of the root lattice node R.

```
void standard_lattice(double *value, double greeks[], double S0,_
                      double X, double sigma, double T, double r,
                      double q, long put, long is_american, long M, long *iflag)
{
/* Input parameters:
   =================
   S0            - the current price of the underlying asset
   X             - the strike price
   sigma         - the volatility
   T             - the time to maturity
   r             - the interest rate
   q             - the continuous dividend yield
   put           - if put is 0 then a call option, otherwise a put option
   is_american   - if is_american is 0 then a European option, otherwise an American option
   M             - the number of time steps
   Output parameters:
   ==================
   value         - the value of the option,
   greeks[]      - the hedge statistics output as follows: greeks[0] is gamma, greeks[1]_
                   is delta, greeks[2] is theta,
   iflag         - an error indicator.
*/
      . . .
    /* Allocate the arrays s[2*M+1], and v[M+1] */
```

Code excerpt 5.11.

```
dt = T/(double)M;
t1 = sigma*sqrt(dt);
u = exp(t1);
d = exp(-t1);
a = exp((r-q)*dt);
p = (a - d)/(u - d);
if ((p < zero) || (p > 1.0)) printf ("Error p out of range\n");
discount = exp(-r*dt);
p_u = discount*p;
p_d = discount*(1.0-p);

/* assign the 2*M+1 asset values */
s[M] = S0;
for (i = 1; i <= M; ++i) {
   s[M+i] = u*s[M+i-1];
   s[M-i] = d*s[M-i+1];
}
/* Find out if the number of time steps, M, is odd or even */
if (((M+1)/2) == (M/2)) { /* then M is even */
   if (put)
     v[M/2] = MAX(X - s[M],zero);
   else
      v[M/2] = MAX(s[M]-X,zero);
}
/* Calculate the option values at maturity  */
P1 = 2*M;
P2 = 0;
for (i = 0; i < (M+1)/2; ++i) {
  if (put) {
     v[M-i] = MAX(X - s[P1],zero);
     v[i]   = MAX(X - s[P2],zero);
  }
  else {
     v[M-i] = MAX(s[P1]-X,zero);
     v[i]   = MAX(s[P2]-X,zero);
  }
  P1 = P1 - 2;
  P2 = P2 + 2;
}
/* now work backwards through the lattice to calculate the current option value */
 P2 = 0;
 for (m1 = M-1; m1 >= 2; --m1) {
   P2 = P2 + 1;
   P1 = P2;
   for (n =0; n <= m1; ++n) {
     if ((v[n] == zero) && (v[n+1] == zero))  {
          hold = zero;
     }
     else
          hold = p_d*v[n] + p_u*v[n+1];
      if (is_american) {
        if (put)
          v[n] = MAX(hold,X-s[P1]);
        else
          v[n] = MAX(hold,s[P1]-X);
      }
      else
        v[n] = hold;
      P1 = P1 + 2;
    }
}
/* The values v[0], v[1] & v[2] correspond to the nodes for m1 = 2, v[3] & v[4] correspond_
   the nodes for m1 = 1 and the
   option value (*value) is the node for m1 = 0, v[5]. For a given time step v[0]_
   corresponds to the lowest asset price,
   v[1] to the next lowest etc.. */

 jj = 2;
 for (m1 = 2; m1 >= 1; --m1) {
   ind = M-m1+1;
   for (n =0; n < m1; ++n) {
      hold =  p_d*v[5-jj-m1-1] + p_u*v[5-jj-m1];
      if (is_american) {
        if (put)
          v[5-jj] = MAX(hold,X-s[ind]);
        else
```

Code excerpt 5.11 (*Continued*).

```
            v[5-jj] = MAX(hold,s[ind]-X);
          }
          else
            v[5-jj] = hold;
            --jj;
            ind = ind + 2;
        }
    }
    *value = v[5];
    if(greeks) {
        /* assign the value of delta (obtained from m1 = 1) */
        greeks[1] = (v[4]-v[3])/(s[M+1]-s[M-1]);
       /*  assign the value of gamma (use the values at time step m1 = 2)  */
        dv1 = v[2] - v[1];
        ds1 = s[M+2] - s[M];
        dv2 = v[1] - v[0];
        ds2 = s[M] - s[M-2];
        h   = 0.5*(s[M+2] - s[M-2]);
        greeks[0] = ((dv1/ds1) - (dv2/ds2))/h;
        /* assign the value of theta */
        greeks[2] = (v[1]-*value)/(2.0*dt);  /* can also write:y greeks[2] = (v[1]-v[5])/_
        (2.0*dt); */
    }
```

Code excerpt 5.11 Function to compute the value of an option using a standard binomial lattice.

5.3.3 Binomial lattice with a control variate

The control variate technique can be used to enhance the accuracy that a standard binomial lattice gives for the value of an American vanilla option. It involves using the same standard binomial lattice to value both an American option and also the equivalent European option. The Black–Scholes formula is then used to compute the accurate value of the European option. If we assume that the error in pricing the European option is the same as that for the American option, we can achieve an improved estimate for the value of the American option.

When applied to the valuation of an American put option this can be expressed as follows:

$$\text{European pricing error, } \Delta_{\mathrm{E}} = p^{\mathrm{BS}}(S, E, \tau) - p^{\mathrm{L}}(S, E, \tau)$$
$$\text{American pricing error, } \Delta_{\mathrm{A}} = P^{*}(S, E, \tau) - P^{\mathrm{L}}(S, E, \tau)$$

where as usual S is the current value of the asset, E is the strike price, and τ is the maturity of the option. Also $p^{\mathrm{BS}}(S, E, \tau)$ is the Black–Scholes value of the European put option, $p^{\mathrm{L}}(S, E, \tau)$ is the binomial lattice estimate of the European put option, $P^{*}(S, E, \tau)$ is the (*unknown*) accurate value of the American put option, and $P^{\mathrm{L}}(S, E, \tau)$ is the binomial lattice estimate of the American put option.

Letting $\Delta_{\mathrm{E}} = \Delta_{\mathrm{A}}$ we then have:

$$p^{\mathrm{BS}}(S, E, \tau) - p^{\mathrm{L}}(S, E, \tau) = P^{*}(S, E, \tau) - P^{\mathrm{L}}(S, E, \tau)$$

which on rearrangement yields:

$$P^{*}(S, E, \tau) = p^{\mathrm{BS}}(S, E, \tau) - p^{\mathrm{L}}(S, E, \tau) + P^{\mathrm{L}}(S, E, \tau)$$

We thus use $P^{*}(S, E, \tau)$ as the improved, control variate estimate for the value of the American put option. Of course, exactly the same approach can be used to obtain an improved estimate for the value of an American call.

Code excerpt 5.12 shows the use of the control variate technique in a standard binomial lattice to provide improved estimates for both the value and the hedge statistics of an American option.

```
/* Set up the arrays as in the standard lattice */
   .   .   .
for (i = 0; i < (M+1)/2; ++i) { /* Calculate the option values at maturity  */
  if (put) {
     a_v[M-i] = MAX(X - s[P1],zero);
     a_v[i]   = MAX(X - s[P2],zero);
  }
  else {
     a_v[M-i] = MAX(s[P1]-X,zero);
     a_v[i]   = MAX(s[P2]-X,zero);
  }
  e_v[i]   = a_v[i];
  e_v[M-i] = a_v[M-i];
  P1 = P1 - 2;
  P2 = P2 + 2;
}
/* now work backwards through the lattice to calculate the current option value */
 P2 = 0;
 for (m1 = M-1; m1 >= 2; --m1) {
   P2 = P2 + 1;
   P1 = P2;
   for (n =0; n <= m1; ++n) {
      if ((a_v[n] == zero) && (a_v[n+1] == zero))
          hold = zero;
      else
          hold = p_d*a_v[n] + p_u*a_v[n+1];
      if (put)
        a_v[n] = MAX(hold,X-s[P1]);
      else
        a_v[n] = MAX(hold,s[P1]-X);
      if ((e_v[n] == zero) && (e_v[n+1] == zero))
          e_v[n] = zero;
      else
          e_v[n] = p_d*e_v[n] + p_u*e_v[n+1];
      P1 = P1 + 2;
   }
 }
/* The American values are stored in the array a_v, and the European values in the array_
   e_v. The array
   indexing is the same as for the standard lattice */

 jj = 2;
 for (m1 = 2; m1 >= 1; --m1) {
   ind = M-m1+1;
   for (n =0; n < m1; ++n) {
      hold =  p_d*a_v[5-jj-m1-1] + p_u*a_v[5-jj-m1];
      if (put)
        a_v[5-jj] = MAX(hold,X-s[ind]);
      else
        a_v[5-jj] = MAX(hold,s[ind]-X);
      e_v[5-jj] = p_d*e_v[5-jj-m1-1] + p_u*e_v[5-jj-m1];
      --jj;
      ind = ind + 2;
   }
}
/* v1 = American binomial approximation, v2 = European Binomial approximation, temp =_
 exact (European) Black-Scholes value */
 black_scholes(&temp,bs_greeks,S0,X,sigma,T,r,q,put,&iflagx);
 *value = (a_v[5] - e_v[5]) + temp; /* return the control variate approximation */
 if(greeks) {
     /* assign the value of delta (obtained from m1 = 1) */
     a_delta = (a_v[4]-a_v[3])/(s[M+1]-s[M-1]);
     e_delta = (e_v[4]-e_v[3])/(s[M+1]-s[M-1]);
     greeks[1] = a_delta - e_delta + bs_greeks[1];
    /*  assign the value of gamma (use the values at time step m1 = 2)  */
     dv1 = a_v[2] - a_v[1];
```

Code excerpt 5.12.

```
    ds1 = s[M+2] - s[M];
    dv2 = a_v[1] - a_v[0];
    ds2 = s[M] - s[M-2];
    h   = 0.5*(s[M+2] - s[M-2]);
    a_gamma = ((dv1/ds1) - (dv2/ds2))/h;
    dv1 = e_v[2] - e_v[1];
    dv2 = e_v[1] - e_v[0];
    e_gamma = ((dv1/ds1) - (dv2/ds2))/h;
    greeks[0] = (a_gamma - e_gamma) + bs_greeks[0];
    /* assign the value of theta */
    a_theta = (a_v[1]-a_v[5])/(2.0*dt);
    e_theta = (e_v[1]-e_v[5])/(2.0*dt);
    greeks[2] = (a_theta - e_theta) + bs_greeks[2];
}
```

Code excerpt 5.12 Function to compute the value and hedge statistics of an American option using a binomial lattice with a control variate.

Finally we should mention that the control variate technique does not just apply to American vanilla options. The method is quite general and can be used to obtain improved estimates for any integral (or exotic option) so long as an accurate (closed form) solution of a *similar* integral is known. One common use of the control variate method is to improve the accuracy of Monte Carlo estimates.

5.3.4 The Binomial lattice with BBS and BBSR

Here we consider the Binomial Black–Scholes (BBS) method and also the Binomial Black–Scholes method with Richardson extrapolation (BBSR) (see Broadie and DeTemple (1996)). As with the control variate method discussed in the previous section, both of these techniques can be used in conjunction with a standard binomial lattice to improve the computed results.

We will first discuss the BBS method.

The BBS method

The BBS method is identical to the standard binomial lattice except that in the last time step (that is just before option maturity) the Black–Scholes formula is used to calculate the option values at maturity. For an n time step binomial lattice this involves evaluating the Black–Scholes formula at each of the n nodes in the penultimate time step; see Fig. 5.1. In Code excerpt 5.13 we define the function `bs_lattice` which incorporates the BBS method into a standard binomial lattice. The reader will have noticed that `bbs_lattice` is *rather lax* concerning the amount of storage that is required; see Section 5.3.2. It uses an array of size $\mathcal{LN}_n$ rather than $\mathcal{LS}_n$ to store the lattice asset prices; the modification to use an array of size $\mathcal{LS}_n$ is left as an exercise.

The benefits of using the BBS approach to price an American call are illustrated in Fig 5.3. Here we compare the results obtained using the function `bbs_lattice` with those computed by the function `standard_ lattice`, the standard binomial lattice of Code excerpt 5.11. It can be clearly seen that the BBS method is significantly more accurate than the standard binomial lattice approach, in which option pricing error exhibits pronounced oscillations.

```
void bbs_lattice(double *value, double greeks[], double S0, double X, double sigma, double T,_
            double r,
            double q, long put, long M, long *iflag)
{
/* Input parameters:
   ================
   S0            - the current price of the underlying asset
   X             - the strike price
   sigma         - the volatility
   T             - the time to maturity
   r             - the interest rate
   q             - the continuous dividend yield
   put           - if put is 0 then a call option, otherwise a put option
   M             - the number of time steps
   Output parameters:
   ==================
   value         - the value of the option, greeks[] - the hedge statistics output as follows:_
                   greeks[0] is gamma,
                   greeks[1] is delta, greeks[2] is theta,
   iflag         - an error indicator.
*/
        .  .  .
          /* allocate the arrays s[((M+2)*(M+1))/2], and v[M+1] */

    dt = T/(double)M;
    t1 = sigma*sqrt(dt);
    u = exp(t1);
    d = exp(-t1);
    a = exp((r-q)*dt);
    p = (a - d)/(u - d);
    if ((p < zero) || (p > 1.0)) return; /* Invalid probability */
    discount = exp(-r*dt);
    p_u = p*discount;
        p_d = (1.0-p)*discount;
    jj = 0;
    s[0] = S0;
    /* The "higher" the value of jj, at a given time instant, the lower the  value of the_
    asset price */
    for (m1 = 1; m1 <= M-1; ++m1) { /* Calculate asset values up to (M-1)th time step */
       for (n = m1; n >= 1; --n) {
          ++jj;
          s[jj] =  u*s[jj-m1];
        }
       ++jj;
       s[jj] = d*s[jj-m1-1];
    }
    for (n = 0; n <= M-1; ++n) { /* Use Black-Scholes for the final step */
       black_scholes(&temp,NULL,s[jj],X,sigma,dt,r,q,put,&iflagx);
       v[n] = temp;
       --jj;
    }
     for (m1 = M-1; m1 >= 3; --m1) { /* work backwards through the lattice */
       for (n =0; n < m1; ++n) {
          if ((v[n] == zero) && (v[n+1] == zero))  {
              hold = zero;
          }
          else
              hold = p_d*v[n] + p_u*v[n+1];
          if (is_american) {
            if (put)
              v[n] = MAX(hold,X-s[jj]);
            else
              v[n] = MAX(hold,s[jj]-X);
          }
          else
            v[n] = hold;
          --jj;
        }
    }
  /* The values v[0], v[1] & v[2] correspond to the nodes for m1 = 2, v1 & v2 correspond to_
     the nodes for m1 = 1 and the
     option value (*value) is the node for m1 = 0. For a given time step v[0] corresponds to_
     the lowest asset price,
      v[1] to the next lowest etc.. */
```

Code excerpt 5.13.

```
    hold = p_d*v[0] + p_u*v[1];
    if (is_american) {
      if (put)
         v1 = MAX(hold,X-s[jj]);
      else
         v1 = MAX(hold,s[jj]-X);
    }
    else
      v1 = hold;
     --jj;
    hold = p_d*v[1] + p_u*v[2];
    if (is_american) {
      if (put)
         v2 = MAX(hold,X-s[jj]);
      else
         v2 = MAX(hold,s[jj]-X);
    }
    else
      v2 = hold;
     --jj;
    hold = p_d*v1 + p_u*v2;
    if (is_american) {
      if (put)
        *value = MAX(hold,X-s[0]);
      else
        *value = MAX(hold,s[0]-X);
    }
    else
      *value = hold;
    if(greeks) {
        /* assign the value of delta (obtained from m1 = 1) */
        greeks[1] = (v2-v1)/(s[1]-s[2]);
       /*  assign the value of gamma (use the values at time step m1 = 2)  */
        dv1 = v[2] - v[1];
        ds1 = s[3] - s[4];
        dv2 = v[1] - v[0];
        ds2 = s[4] - s[5];
        h   = 0.5*(s[3] - s[5]);
        greeks[0] = ((dv1/ds1) - (dv2/ds2))/h;
        /* assign the value of theta */
        greeks[2] = (v[1]-*value)/(2.0*dt);
    }
}
```

Code excerpt 5.13 The function `bbs_lattice` which incorporates the BBS method into a standard binomial lattice. The Black–Scholes formula is evaluated by using the function `black_scholes`, given in Code excerpt 4.1.

The BBSR method

The BBSR method applies two point Richardson extrapolation to the computed BBS values; for more information concerning Richardson extrapolation see Marchuk and Shaidurov (1983). In this method the option price estimates from two BBS lattices, with differing numbers of time steps, are combined to form an improved estimate.

Here we use the following BBSR scheme to compute the value of an American call option:

$$C_{\mathrm{BBSR}}(S, E, \tau, 2n) = \frac{4}{3} C_{\mathrm{BBS}}(S, E, \tau, 2n) - \frac{1}{3} C_{\mathrm{BBS}}(S, E, \tau, n) \qquad (5.3.24)$$

where S is the current asset value, E is the strike price, τ is the option maturity, $C_{\mathrm{BBS}}(S, E, \tau, n)$ is the value of the call option computed using a BBS lattice with n time steps, $C_{\mathrm{BBS}}(S, E, \tau, 2n)$ is the value of the call option computed using a

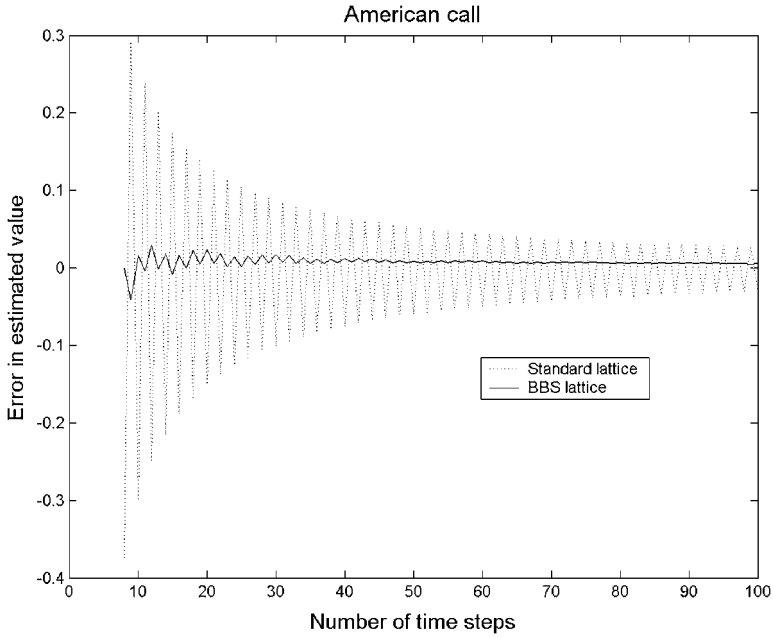

Figure 5.3 The error in the estimated value, *est_val*, of an American call using both a standard binomial lattice and BBS binomial lattice. The parameters used were: $T = 1.0$, $S = 105.0$, $E = 105.0$, $r = 0.1$, $q = 0.02$, $\sigma = 0.3$. The very accurate value (*acc_val*) was 16.1697, and was computed using a 6000 step standard binomial lattice. The error in the estimated value was obtained as *est_val* − *acc_val*.

BBS lattice with $2n$ time steps, and $C_{\text{BBSR}}(S, E, \tau, 2n)$ is the BBSR estimate. We compute the value of an American put using:

$$P_{\text{BBSR}}(S, E, \tau, 2n) = \frac{4}{3} P_{\text{BBS}}(S, E, \tau, 2n) - \frac{1}{3} P_{\text{BBS}}(S, E, \tau, n) \qquad (5.3.25)$$

Figure 5.4 displays the computed BBSR results for an American call option with $S = 105.0$, $\tau = 1.0$, $E = 105.0$, $q = 0.02$ and $\sigma = 0.3$.

In Tables 5.5 and 5.6 the errors in computing both an American put and an American call option are presented; the methods used are the standard binomial lattice, the BBS lattice and the BBSR lattice. It can be seen that the BBSR lattice gives the most accurate results. This is not surprising since, from Eqs. (5.3.24) and (5.3.25) we see that when we use either an n time step standard binomial lattice or an n time step BBS lattice the corresponding BBSR estimate is obtained using both an n time step BBS lattice and also a $2n$ time step BBS lattice. One way of checking whether Richardson extrapolation is providing increased accuracy is to compare the results for a $2n$ time step BBS latticewith those for an n time step BBSR lattice. Inspection of the results shows that Richardson extrapolation has in fact led to an improvement. For example, in Table 5.5 the error for a `160`

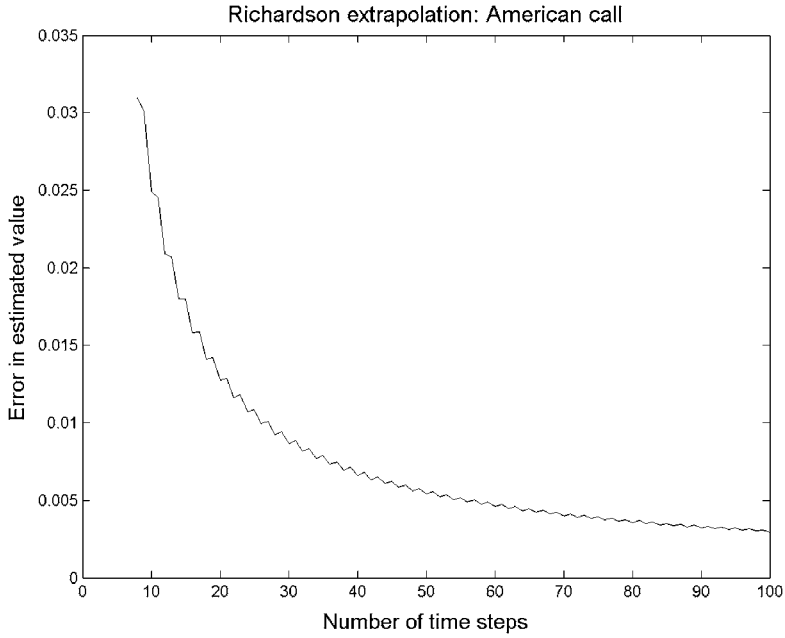

Figure 5.4 The error in the estimated value, *est_val*, of an American call, using a BBSR binomial lattice. The parameters used were: $T = 1.0$, $S = 105.0$, $E = 105.0$, $r = 0.1$, $q = 0.02$, $\sigma = 0.3$. The very accurate value (*acc_val*) was 16.1697, and was computed using a 6000 step standard binomial lattice. The error in the estimated value was obtained as *est_val* − *acc_val*.

time step BBS lattice is `5.0869e-003`, while that for an `80` time step BBSR lattice is `3.5725e-003`; in Table 5.6 the error for an `80` time step BBS lattice is `6.3858e-003`, and that for a `40` time step BBSR lattice is `3.5725e-003`.

5.4 Grid methods for vanilla options

5.4.1 Introduction

In Section 5.3 we discussed the use of binomial lattice methods for valuing both European and American options. The lattice methods we described have the advantage that they are fairly easy to implement and can value simple options, such as vanilla puts and calls, *reasonably* accurately. The use of up and down jump probabilities at the lattice nodes is also an appealing feature, since they are directly related to the stochastic process which is being modelled. However, lattice techniques have the following drawbacks:

- They require small time steps to ensure numerical stability

Table 5.5 The pricing errors for an American call option computed by: a standard binomial lattice, a BBS lattice and also a BBSR lattice

n steps	Standard lattice	BBS lattice	BBSR lattice
20	−1.5075e−001	3.6187e−002	1.2754e−002
30	−1.0057e−001	2.4526e−002	8.6771e−003
40	−7.5382e−002	1.8612e−002	6.6361e−003
50	−6.0244e−002	1.5036e−002	5.4109e−003
60	−5.0141e−002	1.2639e−002	4.5939e−003
70	−4.2919e−002	1.0922e−002	4.0103e−003
80	−3.7499e−002	9.6302e−003	3.5725e−003
90	−3.3282e−002	8.6236e−003	3.2320e−003
100	−2.9908e−002	7.8171e−003	2.9596e−003
110	−2.7146e−002	7.1565e−003	2.7367e−003
120	−2.4844e−002	6.6053e−003	2.5509e−003
130	−2.2896e−002	6.1385e−003	2.3938e−003
140	−2.1226e−002	5.7382e−003	2.2590e−003
150	−1.9778e−002	5.3909e−003	2.1423e−003
160	−1.8511e−002	5.0869e−003	2.0401e−003
170	−1.7393e−002	4.8186e−003	1.9500e−003
180	−1.6399e−002	4.5799e−003	1.8698e−003
190	−1.5510e−002	4.3663e−003	1.7981e−003
200	−1.4710e−002	4.1740e−003	1.7336e−003

The pricing error is defined as *estimated value* − *accurate value*, where the accurate value, 16.1697, was obtained by using a 6000 step standard binomial lattice. The option parameters used were: $T = 1.0$, $S = 105.0$, $E = 105.0$, $r = 0.1$, $q = 0.02$, and $\sigma = 0.3$.

- There is little control over where the lattice nodes are located. This can lead to very poor accuracy when valuing certain types of options; for example, those with barriers at particular asset prices.

One method of avoiding these limitations is through the use of finite-difference grids. Although this approach no longer has the probabilistic interpretation of the binomial lattice it has the following advantages:

- Fewer time steps are required to ensure numerical stability
- There is complete control over the placement of grid lines, and their associated grid nodes.

5.4.2 Uniform grids

The Black–Scholes equation for the value of an option f is given by:

$$\frac{\partial f}{\partial t} + (r - q)S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \tag{5.4.1}$$

We want to solve this equation over the duration of the option, that is from the current time t to the maturity of the option at time $t + \tau$. To do this we

Table 5.6 The pricing errors for an American put option computed by: a standard binomial lattice, a BBS lattice and also a BBSR lattice

n steps	Standard lattice	BBS lattice	BBSR lattice
20	−6.1971e−002	2.3917e−002	7.6191e−003
30	−4.1648e−002	1.6800e−002	6.0465e−003
40	−3.2264e−002	1.1694e−002	4.6165e−003
50	−2.6538e−002	8.4790e−003	4.2654e−003
60	−2.1069e−002	8.7348e−003	3.2946e−003
70	−1.8298e−002	7.2743e−003	2.9633e−003
80	−1.5885e−002	6.3858e−003	2.6088e−003
90	−1.3977e−002	5.9417e−003	2.2099e−003
100	−1.2612e−002	5.3188e−003	2.1793e−003
110	−1.1338e−002	4.9652e−003	2.0992e−003
120	−1.0239e−002	4.6547e−003	1.8723e−003
130	−9.5208e−003	4.1505e−003	1.8808e−003
140	−8.6142e−003	4.0411e−003	1.7505e−003
150	−8.2382e−003	3.6020e−003	1.7341e−003
160	−7.5811e−003	3.5531e−003	1.6411e−003
170	−7.1097e−003	3.3726e−003	1.5507e−003
180	−6.7887e−003	3.1428e−003	1.5478e−003
190	−6.3033e−003	3.1345e−003	1.4134e−003
200	−6.0276e−003	2.9642e−003	1.3973e−003

The pricing error is defined as *estimated value − accurate value*, where the accurate value, 9.2508, was obtained by using a 6000 step standard binomial lattice. The option parameters used were: $T = 1.0$, $S = 105.0$, $E = 105.0$, $r = 0.1$, $q = 0.02$, and $\sigma = 0.3$.

will use a grid in which the asset price S takes n_s uniformly spaced values, $S_j = j\Delta S$, $j = 0, \ldots, n_s - 1$, where ΔS is the spacing between grid points. If $S_{\max}$ is the maximum asset value we want to represent, then the grid spacing, ΔS^*, can be simply calculated as:

$$\Delta S^* = \frac{S_{\max}}{n_s - 1} \tag{5.4.2}$$

However, since we would like to solve the option values and Greeks at the current asset price S_0, we would also like an asset grid line to coincide with the current asset price. This avoids the use of interpolation which is necessary when the asset value does not correspond to a grid line. The method by which we achieve this is outlined in Code excerpt 5.12. Here the user supplies the function `opt_gfd` with values for $S_{\max}$ and $n_s - 1$ from which ΔS^* is computed using Eq. (5.4.2). We then find the integer, n_1, that is just below (or equal to) the value $S_0/\Delta S^*$, and use this to obtain a new grid spacing $\Delta S = S_0/n_1$. This leads to the new asset price discretization $S_j = j\Delta S$, $j = 0, \ldots, n_s - 1$, where we have now ensured that $S_{n_1} = S_0$.

The user also supplies the function `opt_gfd` with the number of time intervals for the grid. When there are n_t time intervals, the grid has $n_t + 1$ uniformly spaced time instants, $t_i = i\Delta t, i = 0, \ldots, n_t$, and the time step is simply:

$$\Delta t = \frac{\tau}{n_t} \tag{5.4.3}$$

As with the binomial lattice methods we will solve the equation backwards in time from maturity (at time $t + \tau$) to the present (time t). So as we solve the equation the time index will start at $i = n_t$ (time $t + \tau$) and decrease to $i = 0$ (current time t).

Here we discuss the grid method of solving the Black–Scholes equation in terms of:

- The finite-difference approximation
- The boundary conditions
- Computation of the option values at a given time instant
- Backwards iteration and early exercise

Each of these aspects will now be considered in turn.

The finite-difference approximation

The option value corresponding to the grid node at which $t_i = i\Delta t$ and $S_j = j\Delta S$ will be denoted by $f_{i,j}$. We will approximate the partial derivative of $f_{i,j}$ with respect to time simply as:

$$\frac{\partial f}{\partial t} = \frac{f_{i+1,j} - f_{i,j}}{\Delta t} \tag{5.4.4}$$

For the other terms in Eq. (5.4.1) we will use the weighted, Θ_m, method. This technique involves selecting an appropriate choice for Θ_m in the range $0 \leqslant \Theta_m \leqslant 1$ so that the *contribution* from node (i, j) is a weighted sum involving the values at nodes (i, j) and $(i+1, j)$. For instance, the term $rf|_{i,j}$ in Eq. (5.4.1) is approximated as:

$$rf|_{i,j} = r\left\{\Theta_m f_{i+1,j} + (1 - \Theta_m) f_{i,j}\right\} \tag{5.4.5}$$

and the term $\frac{\partial f}{\partial S}|_{i,j}$ in Eq. (5.4.1) is approximated as:

$$\left.\frac{\partial f}{\partial S}\right|_{i,j} = \left\{\Theta_m \left.\frac{\partial f}{\partial S}\right|_{i+1,j} + (1 - \Theta_m) \left.\frac{\partial f}{\partial S}\right|_{i,j}\right\} \tag{5.4.6}$$

Using this method we thus obtain, at node (i, j), the following discretized version of Eq. (5.4.1):

$$\begin{aligned} &\frac{f_{i+1,j} - f_{i,j}}{\Delta t} + (r - q) S_j \left\{\Theta_m f'_{i+1,j} + \Theta^*_m f'_{i,j}\right\} \\ &\quad + \frac{1}{2}\sigma^2 S_j^2 \left\{\Theta_m f''_{i+1,j} + \Theta^*_m f''_{i,j}\right\} = r\left\{\Theta_m f_{i+1,j} + \Theta^*_m f_{i,j}\right\} \end{aligned} \tag{5.4.7}$$

where for compactness we have written $\Theta_m^* = 1 - \Theta_m$, and denote the partial derivatives w.r.t. S at node (i, j) as: $f'_{i,j} = \frac{\partial f}{\partial S}|_{i,j}$ and $f''_{i,j} = \frac{\partial^2 f}{\partial S^2}|_{i,j}$.

Finite-difference approximations for these derivatives can be obtained by considering a Taylor expansion about the point $f_{i,j}$. We proceed as follows:

$$f_{i,j+1} = f_{i,j} + f'_{i,j}\Delta S + \frac{1}{2}f''_{i,j}(\Delta S)^2 \tag{5.4.8}$$

$$f_{i,j-1} = f_{i,j} - f'_{i,j}\Delta S + \frac{1}{2}f''_{i,j}(\Delta S)^2 \tag{5.4.9}$$

Subtracting Eq. (5.4.9) from Eq. (5.4.8) we obtain:

$$f_{i,j+1} - f_{i,j-1} = 2f'_{i,j}\Delta S$$

and so

$$f'_{i,j} = \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S} \tag{5.4.10}$$

Adding Eqs. (5.4.9) and (5.4.8) we obtain:

$$f_{i,j+1} + f_{i,j-1} = 2f_{i,j} + f''_{i,j}\Delta S^2$$

which gives:

$$f''_{i,j} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta S^2} \tag{5.4.11}$$

The complete finite-difference approximation to the Black–Scholes equation can then be found by substituting the approximations for the first and second partial derivatives, given in Eqs. (5.4.10) and (5.4.11), into (5.4.7). We thus obtain:

$$\begin{aligned} &r\Delta t\left\{\Theta_m f_{i+1,j} + \Theta_m^* f_{i,j}\right\} \\ &\quad = f_{i+1,j} - f_{i,j} + \frac{(r-q)j\Delta t\mathcal{A}_1}{2} + \frac{\sigma^2 j^2 \Delta t\mathcal{A}_2}{2} \end{aligned} \tag{5.4.12}$$

where we have used the fact that $S_j = j\Delta S$, and for compactness have defined the terms:

$$\mathcal{A}_1 = \Theta_m f_{i+1,j+1} - \Theta_m f_{i+1,j-1} + \Theta_m^* f_{i,j+1} - \Theta_m^* f_{i,j-1}$$

and

$$\begin{aligned} \mathcal{A}_2 = {} & \Theta_m f_{i+1,j+1} + \Theta_m f_{i+1,j-1} - 2\Theta_m f_{i+1,j} + \Theta_m^* f_{i,j+1} \\ & + \Theta_m^* f_{i,j-1} - 2\Theta_m^* f_{i,j} \end{aligned}$$

Collecting like terms in $f_{i,j}$, $f_{i+1,j}$, etc. results in:

$$\begin{aligned} &\mathcal{B}_1 f_{i,j-1} + \mathcal{B}_2 f_{i,j} + \mathcal{B}_3 f_{i,j+1} + \mathcal{C}_1 f_{i+1,j-1} + \mathcal{C}_2 f_{i+1,j} \\ &\quad + \mathcal{C}_3 f_{i+1,j+1} = 0 \end{aligned} \tag{5.4.13}$$

where

$$\mathcal{B}_1 = \frac{-\Theta_m^*(r-q)j\Delta t}{2} + \frac{\Theta_m^*\sigma^2 j^2\Delta t}{2}$$
$$\mathcal{B}_2 = -1 - r\Delta t\Theta_m^* - \Theta_m^*\sigma^2 j^2\Delta t$$
$$\mathcal{B}_3 = \frac{\Theta_m^*(r-q)j\Delta t}{2} + \frac{\Theta_m^*\sigma^2 j^2\Delta t}{2}$$
$$\mathcal{C}_1 = \frac{\Theta_m\sigma^2 j^2\Delta t}{2} - \frac{\Theta_m(r-q)j\Delta t}{2}$$
$$\mathcal{C}_2 = 1 - r\Delta t\Theta_m - \Theta_m\sigma^2 j^2\Delta t$$
$$\mathcal{C}_3 = \frac{\Theta_m(r-q)j\Delta t}{2} + \frac{\Theta_m\sigma^2 j^2\Delta t}{2}$$

Since we are solving the equation backwards in time and we want to determine the option values at time index i from the known option values ($f_{i+1,j+1}$, $f_{i+1,j}$ and $f_{i+1,j-1}$) at time index $i+1$. This can be achieved by rearranging Eq. (5.4.13) as follows:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = R_{i+1,j} \tag{5.4.14}$$

where the right-hand side, $R_{i+1,j}$, is:

$$R_{i+1,j} = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1} \tag{5.4.15}$$

The six coefficients are:

$$a_j = (1-\Theta_m)\frac{\Delta t}{2}\left\{(r-q)j - \sigma^2 j^2\right\} \tag{5.4.16}$$

$$b_j = 1 + (1-\Theta_m)\Delta t\left\{r + \sigma^2 j^2\right\} \tag{5.4.17}$$

$$c_j = -(1-\Theta_m)\frac{\Delta t}{2}\left\{(r-q)j + \sigma^2 j^2\right\} \tag{5.4.18}$$

$$\bar{a}_j = -\Theta_m\frac{\Delta t}{2}\left\{(r-q)j - \sigma^2 j^2\right\} \tag{5.4.19}$$

$$\bar{b}_j = 1 - \Theta_m\Delta t\left\{r + \sigma^2 j^2\right\} \tag{5.4.20}$$

$$\bar{c}_j = \Theta_m\frac{\Delta t}{2}\left\{(r-q)j + \sigma^2 j^2\right\} \tag{5.4.21}$$

For each value of j Eq. (5.4.14) gives us a relationship between three option values, $f_{i+1,j-1}$, $f_{i+1,j}$, $f_{i+1,j+1}$ at time index $i+1$, and three option values $f_{i,j-1}$, $f_{i,j}$, $f_{i,j+1}$ at time index i.

This situation is shown in Fig. 5.5 where we have labelled the grid nodes that contribute to the option value $f_{5,5}$ at grid node E. These are the known option values:

node A: $f_{6,6}$, node B: $f_{6,5}$, and node C: $f_{6,4}$

and the unknown option values

$$\text{node D: } f_{5,6}, \qquad \text{node E: } f_{5,5}, \quad \text{and} \quad \text{node F: } f_{5,4}.$$

Before we solve Eq. (5.4.14), we will briefly consider its characteristics for different values of the weight parameter Θ_m.

When $\Theta_m = 1$ the values of the coefficients in Eq. (5.4.14) are $a_j = c_j = 0$, and $b_j = 1$. This means that Eq. (5.4.14) reduces to:

$$f_{i,j} = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1}$$

This is termed the *explicit method*, and it can be seen that the unknown option value $f_{i,j}$, at the grid node (i, j) is just a weighted sum of the (known) option values $f_{i+1,j-1}$, $f_{i+1,j}$, $f_{i+1,j+1}$. This is the simplest situation to deal with and actually corresponds to a trinomial lattice. However, it has poor numerical properties and usually requires a very small step size to obtain accurate results; see Smith (1985).

When $\Theta_m \neq 1$, the unknown option value $f_{i,j}$ depends not only on the known option values $f_{i+1,j-1}$, $f_{i+1,j}$, $f_{i+1,j+1}$ (as in the explicit method above), but also on the neighboring *unknown* option values $f_{i,j-1}$ and $f_{i,j+1}$. It is now necessary to solve a set of simultaneous equations in order to compute the value $f_{i,j}$. This is therefore called an *implicit method*; see Smith (1985).

The implicit method $\Theta_m = 0$ is also called the *fully implicit method*, since now the unknown value $f_{i,j}$ only depends on the neighboring values $f_{i,j-1}$, $f_{i,j+1}$, and its *previous* value, $f_{i+1,j}$, at time step $i + 1$. This can be shown by substituting $\Theta_m = 0$ in Eqs. (5.4.16)–(5.4.21). We then obtain $\bar{a}_j = \bar{c}_j = 0$, and $\bar{b}_j = 1$, which means that Eq. (5.4.14) reduces to:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = f_{i+1,j}$$

The implicit method $\Theta_m = 0.5$ is also termed the *Crank–Nicolson method*. This method, first used by John Crank and Phyliss Nicolson in 1946 (see Crank and Nicolson (1947)), computes $f_{i,j}$ by giving equal weight to the contributions from time step $i + 1$ and time step i. Substituting $\Theta_m = 0.5$ in Eq. (5.4.16) to Eq. (5.4.21) we obtain the following Crank–Nicolson coefficients:

$$a_j = -\bar{a}_j = \frac{\Delta t}{4}\left\{(r-q)j - \sigma^2 j^2\right\}$$

$$b_j = 1 + \frac{\Delta t}{2}\left\{r + \sigma^2 j^2\right\}$$

$$\bar{b}_j = 1 - \frac{\Delta t}{2}\left\{r + \sigma^2 j^2\right\}$$

$$c_j = -\bar{c}_j = -\frac{\Delta t}{4}\left\{(r-q)j + \sigma^2 j^2\right\}$$

We notice that since we are solving backwards in time, but index time in the forward direction, our values of Θ_m corresponding to implicit and explicit are

different from those normally used. For example, in Smith (1985) $\Theta_m = 0$ is the explicit method and $\Theta_m = 1$ is the implicit method; the Crank–Nicolson method is still $\Theta_m = 0.5$.

The boundary conditions

In order to solve Eq. (5.4.14) at time instant $i\Delta t$ we need to obtain the option values at: the upper asset boundary, the lower asset boundary, and the *initial* values that are specified at option maturity.

Here we calculate the boundary values by using the time independent payoff, p_j, at the jth asset index within the grid. If E is the strike price then vanilla call options have payoffs:

$$p_j = \max(j\Delta S - E, 0), \quad j = 0, \ldots, n_s - 1,$$

and vanilla put options have payoffs:

$$p_j = \max(E - j\Delta S, 0), \quad j = 0, \ldots, n_s - 1$$

Upper asset boundary values

At the upper boundary $j = n_s - 1$ and $(n_s - 1)\Delta S = S_{\max}$; where we note that for the grid to be *useful* we require $S_{\max} > E$.

Here we assume that $S_{\max} > E$ and so for call options:

$$p_{n_s-1} = S_{\max} - E$$

and for put options:

$$p_{n_s-1} = 0$$

The option value at the upper boundary, denoted by f_{BU}, is set to p_{n_s-1}, and we have $f_{i,n_s-1} = f_{\mathrm{BU}}, i = 0, \ldots, n_t$.

Lower asset boundary values

At the lower boundary $j = 0$, and the value of $j\Delta S$ is zero.

So for call options:

$$p_0 = 0$$

and for put options:

$$p_0 = E$$

The option value at the lower boundary, denoted by f_{BL}, is set to p_0, and we have $f_{i,0} = f_{\mathrm{BL}}, i = 0, \ldots, n_t$.

Boundary values at option maturity

At option maturity ($i = n_t$) the initial option (boundary) values are the previously mentioned payouts. If E is the strike price then for vanilla call options:

$$f_{n_t,j} = \max(j\Delta S - E, 0), \quad j = 0, \ldots, n_s - 1,$$

and for vanilla put options:

$$f_{n_t,j} = \max(E - j\Delta S, 0), \quad j = 0, \ldots, n_s - 1$$

This is illustrated in Fig. 5.5 for a vanilla put option with current asset value $S_0 = 20$, strike, $E = 25$, and maturity $\tau = 2$. The grid asset price spacing is $\Delta S = 5$, and the time increment is $\Delta t = 0.2$. At option maturity, corresponding to time index $i = 10$, the value of the put option is zero for all asset indices $j \geqslant 5$.

Computation of the option values at a given time instant

Having found the option boundary values, we are now in a position to solve Eq. (5.4.14) at time instant $t_i = i\Delta t$.

First we note that since $f_{i,0} = f_{\mathrm{BL}}$ and $f_{i,n_s-1} = f_{\mathrm{BU}}$ Eq. (5.4.14) only needs to be solved for values of the asset index j in the range $j = 1$ to $j = n_s - 2$.

We now deal with the following situations:

- *Case* 1: $j = 1$, the asset grid line just above the lower boundary
- *Case* 2: $j = n_s - 2$, the asset grid line just below the upper boundary
- *Case* 3: all other asset grid lines not included in *Case* 1 or *Case* 2

and consider the form that Eq. (5.4.14) takes under each condition.

Case 1: $j = 1$

Substituting $j = 1$ into Eq. (5.4.14) we obtain:

$$a_1 f_{i,0} + b_1 f_{i,1} + c_1 f_{i,2} = \bar{a}_1 f_{i+1,0} + \bar{b}_1 f_{i+1,1} + \bar{c}_1 f_{i+1,2}$$

Now, since $f_{i,0} = f_{\mathrm{BL}}$, this becomes:

$$b_1 f_{i,1} + c_1 f_{i,2} = (\bar{a}_1 - a_1) f_{\mathrm{BL}} + \bar{b}_1 f_{i+1,1} + \bar{c}_1 f_{i+1,2}$$

or equivalently:

$$b_1 f_{i,1} + c_1 f_{i,2} = R_{i+1,1} \tag{5.4.22}$$

where

$$R_{i+1,1} = (\bar{a}_1 - a_1) f_{BL} + \bar{b}_1 f_{i+1,1} + \bar{c}_1 f_{i+1,2} \tag{5.4.23}$$

Case 2: $j = n_s - 2$

Substituting $j = n_s - 1$ into Eq. (5.4.14) we obtain:

$$\begin{aligned} &a_{n_s-2} f_{i,n_s-3} + b_{n_s-2} f_{i,n_s-2} + c_{n_s-2} f_{i,n_s-1} \\ &\quad = \bar{a}_{n_s-2} f_{i+1,n_s-3} + \bar{b}_{n_s-2} f_{i+1,n_s-2} + \bar{c}_{n_s-2} f_{i+1,n_s-1} \end{aligned}$$

Since $f_{i,n_s-1} = f_{\mathrm{BU}}$ this gives:

$$\begin{aligned} &a_{n_s-2} f_{i,n_s-3} + b_{n_s-2} f_{i,n_s-2} \\ &\quad = \bar{a}_{n_s-2} f_{i+1,n_s-3} + \bar{b}_{n_s-2} f_{i+1,n_s-2} + (\bar{c}_{n_s-2} - c_{n_s-2}) f_{\mathrm{BU}} \end{aligned}$$

or equivalently:

$$a_{n_s-2} f_{i,n_s-3} + b_{n_s-2} f_{i,n_s-2} = R_{i+1,n_s-2} \tag{5.4.24}$$

where

$$R_{i+1,n_s-2} = \bar{a}_{n_s-2} f_{i+1,n_s-3} + \bar{b}_{n_s-2} f_{i+1,n_s-2} + (\bar{c}_{n_s-2} - c_{n_s-2}) f_{\mathrm{BU}} \tag{5.4.25}$$

Case 3

In this case the boundary values do not enter into the expressions, and we simply restate Eq. (5.4.14) as:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = R_{i+1,j}, \quad j = 3, \ldots, n_s - 3, \tag{5.4.26}$$

where as before the right-hand side, $R_{i+1,j}$, is:

$$R_{i+1,j} = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1} \tag{5.4.27}$$

We can now gather all the information in Eqs. (5.4.23)–(5.4.27) and represent it by the following tridiagonal system:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & a_{n_s-3} & b_{n_s-3} & c_{n_s-3} \\ 0 & 0 & 0 & 0 & a_{n_s-2} & b_{n_s-2} \end{pmatrix} \begin{pmatrix} f_{i,1} \\ f_{i,2} \\ . \\ . \\ f_{i,n_s-3} \\ f_{i,n_s-2} \end{pmatrix} = \begin{pmatrix} R_{i+1,1} \\ R_{i+1,2} \\ . \\ . \\ R_{i+1,n_s-3} \\ R_{i+1,n_s-2} \end{pmatrix} \tag{5.4.28}$$

In matrix notation Eq. (5.4.28) can be written as:

$$Ax = \mathcal{R} \tag{5.4.29}$$

where A is the $(n_s - 2) \times (n_s - 2)$ tridiagonal matrix containing the known coefficients a_j, $j = 2, \ldots, n_s - 2$, b_j, $j = 1, \ldots, n_s - 2$, and c_j, $j = 1, \ldots, n_s - 3$. The vector $\mathcal{R}$ denotes the known right-hand side, $R_{i+1,j}$, $j = 1, \ldots, n_s - 2$, and the vector x contains the unknown option values that we wish to compute, $f_{i,j}$, $j = 1, \ldots, n_s - 2$.

It is well known that, if matrix A is nonsingular, Eq. (5.4.29) can be solved using an LU decomposition. Here we factorize the $n \times n$ matrix A as:

$$A = LU$$

where L is an $n \times n$ lower triangular matrix with 1s on the diagonal and U is an $n \times n$ upper triangular matrix. We illustrate the LU decomposition for a full 4×4 matrix below:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 \\ l_{3,1} & l_{3,2} & 1 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & 1 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{pmatrix} \tag{5.4.30}$$

If A is a tridiagonal matrix then the LU decomposition takes the simpler form:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} \\ 0 & 0 & a_{4,3} & a_{4,4} \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 \\ 0 & l_{3,2} & 1 & 0 \\ 0 & 0 & l_{4,3} & 1 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & 0 & 0 \\ 0 & u_{2,2} & u_{2,3} & 0 \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{pmatrix} \tag{5.4.31}$$

where it can be seen that now both L and U are bidiagonal.

Once the LU decomposition of A has been found, it is possible to solve for x in Eq. (5.4.29) by using a two stage method (see for example Golub and Van Loan (1989)). Here forward elimination is used to solve $Ly = \mathcal{R}$, and then back-substitution is applied to $Ux = y$. We can thus write the procedure as:

$$Ax = (LU)x = L(Ux) = Ly = \mathcal{R}$$

We will now provide code excerpts which show how to solve the $(n_s - 2) \times (n_s - 2)$ tridiagonal system represented by Eq. (5.4.29). These excerpts are in fact contained within the larger Code excerpt 5.18, which displays the complete C code for the option pricing function `opt_gfd`. If the reader requires more detail concerning the precise code used for option pricing, then this code should be consulted. (It should be noted that in Code excerpt 5.18, time is indexed using j and asset price using index i. We have modified the indices for the smaller code excerpts given below so that, as might be expected, time is indexed using i, and asset price using j. The author apologizes for any inconvenience this may cause.) Here, for brevity, we will assume that all the required arrays have already been allocated and loaded with the relevant information.

First we need to compute the LU decomposition of the tridiagonal matrix A. The code to achieve this is given in Code excerpt 5.14. Here we use the following three arrays to store the elements of the tridiagonal matrix A: array **b** contains the diagonal elements, array **c** contains the upper diagonal elements, and array **a** holds the lower diagonal elements.

It should be noted we do not explicitly compute the elements of the matrix L. This is because all the diagonal elements of L are known to be 1, and the sub-diagonal elements of L can be computed from the diagonal elements of U by using `l[j] = a[j]/u[j-1]`. Also we do not need to compute the upper diagonal elements of U since they are known to be the same as the upper diagonal

```
u[1] = b[1];
if (u[1] == 0.0) printf ("ERROR in array u \n");
for(j=2; j <=ns-2; ++j) {
   u[j] = b[j] - a[j]*c[j-1]/u[j-1];
   if (u[j] == 0.0) printf ("ERROR in array u \n");
}
```

Code excerpt 5.14 Computer code that calculates the diagonal elements of the matrix U, in an LU decomposition of a tridiagonal matrix A. The elements of matrix A are stored in the following arrays: array `b` contains the diagonal elements, array `c` contains the upper diagonal elements, and array `a` holds the lower diagonal elements. The diagonal elements of U are stored in the array `u` for later use, in Code excerpts 5.15 and 5.16.

```
work[1] = rhs[1];
for(j=2; j<=ns-2; ++j) {
   work[j] = rhs[j] - a[j]*work[j-1]/u[j-1];
}
```

Code excerpt 5.15 Computer code that uses forward elimination to solve the lower triangular system $Ly = \mathcal{R}$, where y is stored in the array `work`.

```
opt_vals[ns-2] = work[ns-2]/u[ns-2];
for(j = ns-2; j >= 1; --j)
   opt_vals[j] = (work[j] - c[j]*opt_vals[j+1])/u[j];
```

Code excerpt 5.16 Computer code that uses back-substitution to solve the upper triangular system $Ux = y$. At time instant $t_i = i\Delta t$, the elements of x are the calculated option values $f_{i,j}$, $i = 1, \ldots, n_{s-2}$.

elements of the original matrix A, and are contained in the array `c`; see for example Hager (1988).

Having computed the LU decomposition we can now solve the lower triangular system $Ly = \mathcal{R}$ using forward elimination; this is shown in Code excerpt 5.15.

In Code excerpt 5.15 we make use of the following two arrays: the array `rhs` which is used to store the elements of the right-hand side $\mathcal{R}$, and the array `work` which is both used as workspace and to store the computed solution vector y. As previously mentioned the subdiagonal elements of L are given by `l[j] = a[j]/u[j-1]`. This means that in Code excerpt 5.15, the line:

$$\texttt{work[j]} = \texttt{rhs[j]} - \texttt{a[j]} * \texttt{work[j} - \texttt{1]/u[j} - \texttt{1]};$$

is in fact equivalent to:

$$\texttt{work[j]} = \texttt{rhs[j]} - \texttt{l[j]} * \texttt{work[j} - \texttt{1]};$$

where `l[j],j=2,..,ns-2`, contains the subdiagonal elements of L, if we had (needlessly) decided to allocate space for an extra array called `l`.

We are now in a position to solve the triangular system $Ux = y$ by using back-substitution. The code to achieve this is given in Code excerpt 5.16. Here the array `work` contains the previously computed values of y, the diagonal elements of U are contained in the array `u`, and (as previously mentioned) the upper diagonal elements of U are stored in the array `a`.

In Code excerpt 5.16 the array `opt_vals` contains the solution vector x. As its name suggests the contents of the array `opt_vals` are in fact the computed option values, $f_{i,j}$, $j = 1, \ldots, n_s - 2$, in Eq. (5.4.28) and represent the solution of the Black–Scholes partial differential equation at time instant $t_i = i\Delta t$ based on the previously computed option values $f_{i+1,j}$, $j = 1, \ldots, n_s - 2$.

Backwards iteration and early exercise

The Black–Scholes equation can be solved over the time interval t to $t+\tau$ by iteratively solving Eq. (5.4.28). We iterate backwards in time by solving Eq. (5.4.28) at the ith time step and then using the computed values to solve Eq. (5.4.28) for the $(i-1)$th time step. The option values at current time t are obtained when time index $i = 0$ is reached. It can be seen that the grid method yields $n_s - 2$ option values, $f_{0,j}$, $j = 1, \ldots, n_s - 2$, which correspond to the current asset prices:

$$S_0^j = j\Delta S, \quad j = 1, \ \ldots, n_s - 2$$

As previously mentioned the asset price S_0 coincides with grid index $j = n_1$. Therefore $S_0 = S_0^{n_1}$, and the option value for the current asset price S_0 is given by f_{0,n_1}.

This is in contrast to the lattice methods discussed in Chapter 4, which yield a single option value corresponding to the root node.

The option values obtained using the grid methods we have just described are for vanilla European options. However, vanilla European options can be more accurately valued by using the Black–Scholes option pricing formula discussed in Chapter 4. The importance of finite difference grids is that, by slightly modifying our backward iterative method, we can take into account the possibility of *early exercise*, and thus price American options.

This can be achieved by using Code excerpt 5.17 to modify the option prices contained in the array `opt_vals` as follows:

```
if (put) {  /* a put */
   for(j=1; j<=ns-2; ++j)
       opt_vals[j] = MAX(opt_vals[j],E-s[j]);
   }
   else { /* a call */
   for(j=1; j<=ns-2; ++j)
       opt_vals[j] = MAX(opt_vals[j],s[j]-E);
   }
```

Code excerpt 5.17 Computer code that modifies the computed option values contained in array `opt_vals` to include the possibility of *early exercise*; this is required if we are to determine the value of American options. Here `s[j]` contains the asset value at asset index j, `opt_vals[j]` contains the option value (computed by Code excerpt 5.16) at asset index j, and E is the strike price.

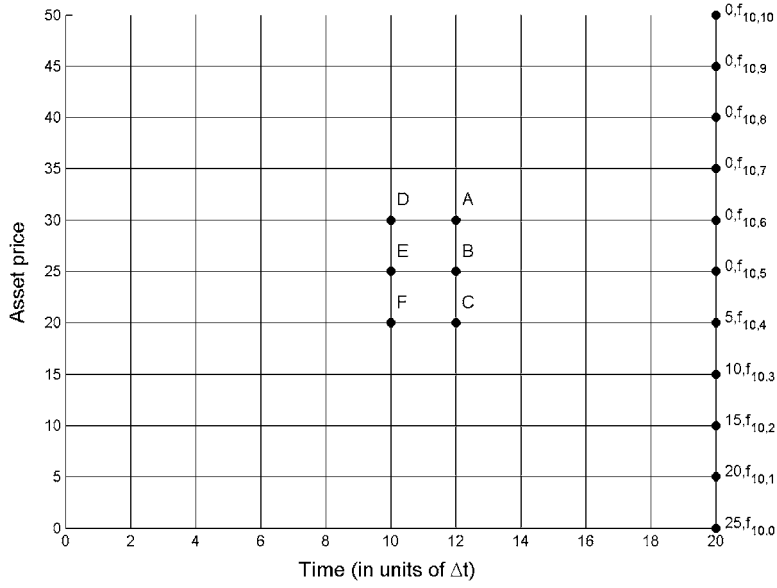

Figure 5.5 An example uniform grid, which could be used to estimate the value of a vanilla option which matures in two years' time. The grid parameters are: $n_s = n_t = 10$, $\Delta t = 0.2$, $\Delta S = 5$, and $S_{\max} = 50$. The option parameters are $E = 25$, $S_0 = 20$, and $\tau = 2.0$. As usual we denote the grid node option values by $f_{i,j}$, where i is the time index and j is the asset index. The option values of the grid nodes at maturity for a vanilla put are thus labelled as val, $f_{10,j}$, $j = 0, \ldots, 10$, where val is the value of the option at the node; these are shown on the right-hand grid boundary. Since $E = 25$ only those nodes with $j < 5$ have nonzero option values.

Now that we know how to solve the Black–Scholes equation it is possible to include, without much difficulty, more *exotic features* such as lock out periods, barriers, rebates, etc.

The routine `opt_gfd` solves the Black–Scholes equation using a uniform grid. The asset price is set to one of the grid lines, which means that interpolation is not required.

5.4.3 Nonuniform grids

In the previous section we showed how to solve the Black–Scholes equation using a uniform grid. Although this approach will provide satisfactory solutions to many option pricing problems, there are situations in which it is important to be able to place grid lines at locations that do not correspond to those available in a uniform grid. Increasing the density of grid lines in regions of interest can lead to improved accuracy in both the estimated option values and also the estimates of the hedge statistics (the Greeks).

Here we provide an example which illustrates the benefits of using a nonuniform grids in the evaluation of down and out call barrier options. Later on in

Section 5.4.6 we give a further example which shows the use of nonuniform grids to evaluate double barrier options.

The purpose of this section is to show how to discretize the Black–Scholes equation using a nonuniform grid, and to derive an expression (see Eq. (5.4.39)) that is equivalent to Eq. (5.4.14). Although the tridiagonal system of equations we have to solve in this section will be different from that in Section 5.4, the solution method is exactly the same. This means that once we have derived Eq. (5.4.39) all the other information which we require to evaluate both European and American options is available in Section 5.4 under the headings:

- The boundary conditions
- Computation of the option values at a given time instant
- Backwards iteration and early exercise.

We will now consider the finite-difference approximation for a nonuniform grid, and then show how to value the down and out call barrier option.

The finite-difference approximation

Here we consider how to discretize the Black–Scholes equation using a nonuniform grid, in which both the asset price interval ΔS and the time step Δt are not constant but can vary throughout the grid.

Allowing for a nonconstant time step is quite simple. The time step occurs in both the first derivative $f_{i,j}$ (see Eq. (5.4.4)) and in the option value equations (see Eqs. (5.4.14)–(5.4.21)) as the constant Δt. To incorporate a varying time step, $\Delta t_i, i = 0, n_t$, thus only requires setting $\Delta t = \Delta t_i$, at the ith time step and then continuing with the solution method outlined in Section 5.4.

The incorporation of nonconstant asset price intervals requires more work. This is because the finite-difference approximations to the first and second derivatives $f'_{i,j}$ and $f''_{i,j}$, in Eqs. (5.4.10) and (5.4.11) are based on a Taylor expansion about the point $f_{i,j}$.

We will now derive expressions for these derivatives.

If we let $\Delta X_j^- = S_j - S_{j-1}$ and $\Delta X_j^+ = S_{j+1} - S_j$ and then use a Taylor expansion about $f_{i+1,j}$ we have

$$f_{i+1,j+1} = f_{i+1,j} + f'_{i+1,j}\Delta X_j^+ + \frac{1}{2} f''_{i+1,j}\left(\Delta X_j^+\right)^2 \tag{5.4.32}$$

and also

$$f_{i+1,j-1} = f_{i+1,j} - f'_{i+1,j}\Delta X_j^- + \frac{1}{2} f''_{i+1,j}\left(\Delta X_j^-\right)^2 \tag{5.4.33}$$

Multiplying Eq. (5.4.32) by ΔX_j^- and adding it to ΔX_j^+ times Eq. (5.4.33), gives

$$\begin{aligned}&\Delta X_j^+ f_{i+1,j-1} + \Delta X_j^- f_{i+1,j+1}\\ &\quad = \Delta X_j^- f_{i+1,j} + \Delta X_j^+ f_{i+1,j} + \frac{1}{2} f''_{i+1,j}\left\{\left(\Delta X_j^+\right)^2 \Delta X_j^- + \left(\Delta X_j^-\right)^2 \Delta X_j^+\right\}\end{aligned}$$

Therefore

$$\frac{1}{2}f''_{i+1,j} = \frac{\Delta X_j^+ f_{i+1,j-1} + \Delta X_j^- f_{i+1,j+1} - \Delta X_j^- f_{i+1,j} - \Delta X_j^+ f_{i+1,j}}{(\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+}$$

So

$$f''_{i+1,j} = \frac{2\{\Delta X_j^+ f_{i+1,j-1} + \Delta X_j^- f_{i+1,j+1} - f_{i+1,j}(\Delta X_j^- + \Delta X_j^+)\}}{(\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+} \tag{5.4.34}$$

To calculate $f'_{i+1,j}$ we rearrange Eq. (5.4.33) to obtain

$$-f'_{i+1,j}\Delta X_j^- = f_{i+1,j-1} - f_{i+1,j} - \frac{1}{2}f''_{i+1,j}\left(\Delta X_j^-\right)^2$$

and

$$f'_{i+1,j} = \frac{f_{i+1,j} - f_{i+1,j-1}}{\Delta X_j^-} + \frac{1}{2}f''_{i+1,j}\Delta X_j^- \tag{5.4.35}$$

If we now substitute for $f''_{i+1,j}$, from Eq. (5.4.34), into Eq. (5.4.35) we have

$$f'_{i+1,j} = \frac{f_{i+1,j} - f_{i+1,j-1}}{\Delta X_j^-} + \frac{\{\Delta X_j^+ f_{i+1,j-1} - (\Delta X_j^- + \Delta X_j^+) f_{i+1,j} + \Delta X_j^- f_{i+1,j+1}\}\Delta X_j^-}{(\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+}$$

which simplifies to give

$$f'_{i+1,j} = \frac{(\Delta X_j^+)^2(f_{i+1,j} - f_{i+1,j-1}) - (\Delta X_j^-)^2 f_{i+1,j} + (\Delta X_j^-)^2 f_{i+1,j+1}}{(\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+}$$

so that we finally have

$$f'_{i+1,j} = \frac{(\Delta X_j^-)^2 f_{i+1,j+1} + ((\Delta X_j^+)^2 - (\Delta X_j^-)^2) f_{i+1,j} - (\Delta X_j^+)^2 f_{i+1,j-1}}{(\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+} \tag{5.4.36}$$

As in Section 5.4, we can now substitute the expressions for $f'_{i+1,j}$ and $f''_{i+1,j}$ given in Eqs. (5.4.36) and (5.4.34), into Eq. (5.4.7) the discretized Black–Scholes equation. If we let $D = (\Delta X_j^+)^2 \Delta X_j^- + (\Delta X_j^-)^2 \Delta X_j^+$ we then obtain

$$r\Delta t\left(\Theta_m f_{i+1,j} + \Theta_m^* f_{i,j}\right) = f_{i+1,j} - f_{i,j} + \frac{(r-q)S_j \Delta t \mathcal{A}_1}{D} + \frac{\sigma^2 S_j^2 \Delta t \mathcal{A}_2}{D} \tag{5.4.37}$$

where $\Theta_m^* = 1 - \Theta_m$, and

$$\begin{aligned}\mathcal{A}_1 = \Theta_m\big[&f_{i+1,j+1}\big(\Delta X_j^-\big)^2 - f_{i+1,j-1}\big(\Delta X_j^+\big)^2 \\ &- f_{i+1,j}\big\{\big(\Delta X_j^-\big)^2 - \big(\Delta X_j^+\big)^2\big\}\big] \\ + \Theta_m^*\big[&f_{i,j+1}\big(\Delta X_j^-\big)^2 - f_{i,j-1}\big(\Delta X_j^+\big)^2 - f_{i,j}\big\{\big(\Delta X_j^-\big)^2 - \big(\Delta X_j^+\big)^2\big\}\big]\end{aligned}$$

and

$$\begin{aligned}\mathcal{A}_2 = \Theta_m\big[&f_{i+1,j+1}\Delta X_j^- + f_{i+1,j-1}\Delta X_j^+ - f_{i+1,j}\big\{\Delta X_j^- + \Delta X_j^+\big\}\big] \\ + \Theta_m^*\big[&f_{i,j+1}\Delta X_j^- + f_{i,j-1}\Delta X_j^+ - f_{i,j}\big\{\Delta X_j^- + \Delta X_j^+\big\}\big]\end{aligned}$$

Collecting like terms, we obtain:

$$\begin{aligned}&\mathcal{B}_1 f_{i,j-1} + \mathcal{B}_2 f_{i,j} + \mathcal{B}_3 f_{i,j+1} + \mathcal{C}_1 f_{i+1,j-1} + \mathcal{C}_2 f_{i+1,j} \\ &\quad + \mathcal{C}_3 f_{i+1,j+1} = 0\end{aligned} \tag{5.4.38}$$

where

$$\mathcal{B}_1 = \frac{-\Theta_m^*(r-q)S_j\Delta t(\Delta X_j^+)^2}{D} + \frac{(1-\theta)\sigma^2 S_j^2 \Delta t \Delta X_j^+}{D}$$

$$\begin{aligned}\mathcal{B}_2 = -1 - r\Delta t\Theta_m^* &- \frac{\Theta_m^*\sigma^2 S_j^2 \Delta t(\Delta X_j^- + \Delta X_j^+)}{D} \\ &- \frac{\Theta_m^*(r-q)S_j\Delta t\{(\Delta X_j^-)^2 - (\Delta X_j^+)^2\}}{D}\end{aligned}$$

$$\mathcal{B}_3 = \frac{\Theta_m^*(r-q)S_j\Delta t(\Delta X_j^-)^2}{D} + \frac{\Theta_m^*\sigma^2 S_j^2\Delta t\Delta X_j^-}{D}$$

$$\mathcal{C}_1 = \frac{\Theta_m\sigma^2 S_j^2\Delta t\Delta X_j^+}{D} - \frac{\Theta_m(r-q)S_j\Delta t(\Delta X_j^+)^2}{D}$$

$$\begin{aligned}\mathcal{C}_2 = 1 - r\Delta t\Theta_m &- \frac{\Theta_m(r-q)S_j\Delta t\{(\Delta X_j^-)^2 - (\Delta X_j^+)^2\}}{D} \\ &- \frac{\Theta_m\sigma^2 S_j^2\Delta t\{\Delta X_j^- + \Delta X_j^+\}}{D}\end{aligned}$$

$$\mathcal{C}_3 = \frac{\Theta_m(r-q)S_j\Delta t(\Delta X_j^-)^2}{D} + \frac{\Theta_m\sigma^2 S_j^2\Delta t\Delta X_j^-}{D}$$

Since we are solving the Black–Scholes equation backwards in time we will rearrange Eq. (5.4.38) as:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j = R_{i+1,j} \tag{5.4.39}$$

where the right-hand side $R_{i+1,j}$ is:

$$R_{i+1,j} = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1} \tag{5.4.40}$$

and the coefficients are

$$a_j = \Theta_m^* \Delta t \left\{ \frac{(r-q)S_j(\Delta X_j^+)^2}{D} - \frac{\sigma^2 S_j^2 \Delta X_j^+}{D} \right\} \tag{5.4.41}$$

$$b_j = 1 + \Delta t \Theta_m^* \left\{ r + \frac{\sigma^2 S_j^2 (\Delta X_j^- + \Delta X_j^+)}{D} + \frac{(r-q)S_j\{(\Delta X_j^-)^2 - (\Delta X_j^+)^2\}}{D} \right\} \tag{5.4.42}$$

$$c_j = \Theta_m^* \Delta t \left\{ \frac{-(r-q)S_j(\Delta X_j^-)^2}{D} - \frac{\sigma^2 S_j^2 \Delta X_j^-}{D} \right\} \tag{5.4.43}$$

$$\bar{a}_j = \Theta_m \Delta t \left\{ \frac{\sigma^2 S_j^2 \Delta X_j^+}{D} - \frac{(r-q)S_j(\Delta X_j^+)^2}{D} \right\} \tag{5.4.44}$$

$$\bar{b}_j = 1 - \Theta_m r \Delta t - \Theta_m \Delta t \left\{ \frac{(r-q)S_j\{(\Delta X_j^-)^2 - (\Delta X_j^+)^2\}}{D} + \frac{\sigma^2 S_j^2 \{\Delta X_j^- + \Delta X_j^+\}}{D} \right\} \tag{5.4.45}$$

$$\bar{c}_j = \Theta_m \Delta t \left\{ \frac{(r-q)S_j(\Delta X_j^-)^2}{D} + \frac{\sigma^2 S_j^2 \Delta X_j^-}{D} \right\} \tag{5.4.46}$$

Here Eq. (5.4.39), as is the case for Eq. (5.4.14) in Section 5.4, provides the relationship between the three option values $f_{i+1,j-1}$, $f_{i+1,j}$, $f_{i+1,j+1}$ at time index $i+1$, and the three option values $f_{i,j-1}$, $f_{i,j}$, $f_{i,j+1}$ at time index i. It can also be seen that Eq. (5.4.39) is the nonuniform grid equivalent of Eq. (5.4.14) given in Section 5.4. We will now show that Eqs. (5.4.39) and (5.4.14) are identical when a uniform grid is used, that is $\Delta X_j^+ = \Delta X_j^-$. We proceed as follows:

Let $\Delta X_j^+ = \Delta X_j^- = \Delta S$ and $S_j = j\Delta S$.

So

$$D = \left(\Delta X_j^+\right)^2 \Delta X_j^- + \left(\Delta X_j^-\right)^2 \Delta X_j^+$$

$$= 2(\Delta S)^3 \frac{(\Delta X_j^+)^2}{D} = \frac{(\Delta X_j^-)^2}{D} = \frac{(\Delta S)^2}{2(\Delta S)^3} = \frac{1}{2\Delta S}$$

$$\frac{\Delta X_j^+}{D} = \frac{\Delta X_j^-}{D} = \frac{1}{2\Delta S^2}$$

$$\frac{(\Delta X_j^+)^2 - (\Delta X_j^-)^2}{D} = 0$$

If we substitute the above values into Eqs. (5.4.41)–(5.4.46) we obtain the following expressions for the coefficients in Eq. (5.4.39).

$$a_j = (1-\Theta_m)\Delta t\left\{\frac{(r-q)S_j}{2\Delta S} - \frac{\sigma^2 S_j^2}{2\Delta S^2}\right\} = (1-\Theta_m)\frac{\Delta t}{2}\left\{(r-q)j - \sigma^2 j^2\right\}$$

$$b_j = 1 + \Delta t(1-\Theta_m)\left\{r + \frac{\sigma^2 S_j^2}{\Delta S^2}\right\} = 1 + (1-\Theta_m)\Delta t\left\{r + \sigma^2 j^2\right\}$$

$$c_j = (1-\Theta_m)\Delta t\left\{\frac{-(r-q)S_j}{2\Delta S} - \frac{\sigma^2 S_j^2}{2\Delta S^2}\right\}$$

$$= -(1-\Theta_m)\frac{\Delta t}{2}\left\{(r-q)j + \sigma^2 j^2\right\}$$

$$\bar{a}_j = \Theta_m \Delta t\left\{\frac{\sigma^2 S_j^2}{2\Delta S^2} - \frac{(r-q)S_j}{2\Delta S}\right\} = -\Theta_m\frac{\Delta t}{2}\left\{(r-q)j - \sigma^2 j^2\right\}$$

$$\bar{b}_j = 1 - \Theta_m r\Delta t - \frac{\Theta_m \sigma^2 S_j^2 \Delta t}{\Delta S^2} = 1 - \Theta_m \Delta t\left\{r + \sigma^2 j^2\right\}$$

$$\bar{c}_j = \Theta_m \Delta t\left\{\frac{(r-q)S_j}{2\Delta S} + \frac{\sigma^2 S_j^2}{\Delta S^2}\right\} = \Theta_m\frac{\Delta t}{2}\left\{(r-q)j + \sigma^2 j^2\right\}$$

It can be seen that these coefficients are identical to those given in Section 5.4.2 Eqs. (5.4.16)–(5.4.21).

We now provide examples of using nonuniform grids to evaluate European down and out call options.

Valuation of a down and out call option

Here the improved accuracy that can be achieved by using nonuniform grids instead of uniform grids is illustrated in Figs. 5.7 and 5.8. The uniform grids are constructed using the method outlined in Section 5.4 and Code excerpt 5.18. That is, an asset grid line is set to coincide with the current asset price S_0, and the other grid lines are positioned above and below S_0 with a uniform spacing of ΔS. The disadvantage of this approach is that there will be an unspecified pricing error that depends on the distance, d_s, of the barrier level, B, to the

```
void opt_gfd(double theta_m, double asset_price, double sigma, double r, double T,
        double strike, long is_american, long put, double *option_value,
        double greeks[], double q, long pns, long nt, double smax, long *iflag)
{
/* Input parameters:
   =================
   theta_m           - the value of theta used for the finite difference method,
   asset_price       - the current price of the underlying asset,
   sigma             - the volatility,
   r                 - the interest rate,
   T                 - the time to maturity,
   strike            - the strike price,
   is_american       - if is_american is 0 then a European option, otherwise an American_
                       option,
```

Code excerpt 5.18.

```
   put               - if put is 0 then a call option, otherwise a put option,
   q                 - the continuous dividend yield,
   pns               - the maximum asset index on the grid, corresponding to the upper_
                       boundary,
   nt                - the number of time intervals,
   smax              - the maximum asset price.
   Output parameters:
   option_value      - the value of the option,
   greeks[]          - the hedge statistics output as follows: greeks[0] is gamma, greeks[1]_
                       is delta, and greeks[2] is theta,
   iflag             - an error indicator.
*/
     double *a,*b,*c,*a1,*b1,*c1,*opt_vals,*vals,*rhs,*s,*work,*u;
     double ds,dt;
     long i,j;
     double tmp,t2,time_2mat;
     long n1,n2,ind=0;
     double sig2,temp[4];

      if (asset_price >= smax) printf ("ERROR asset price >= smax");
      n1 = floor((asset_price/smax)*(double)pns);
      n2 = pns - n1;
      ds = asset_price/(double)n1;
      dt = T/(double)nt;     /* time interval size  */
      ns = n1+n2+1;
/* Note: Now nps = ns-1. Since we define asset grid lines 0...ns-1, this is the maximum grid_
   line; corresponding
   to the upper boundary. The lower boundary is at the asset grid line 0, and we solve for_
   option values between
   the asset grid line 1 and the asset grid line ns-2  */

/* Allocate (all size ns+1) the arrays: a, b, c, a1, b1, c1, opt_vals, vals, rhs, s, work_
   and u */
          .  .  .
      s[0] = 0.0;
      s[n1] = asset_price;
      for(i=1; i<=n1-1; ++i )  /* set prices below asset_price */
         s[i] = (double)i * ds;
      for(i=1; i<= n2+1; ++i )  /* set prices above asset_price */
         s[n1+i] = asset_price + (double)i * ds;

/* Set up the RHS and LHS coefficients a[], b[] and c[] are the LHS coefficients
   for the unknown option values (time step j) a1[], b1[] and c1[] are the values of the
   RHS coefficients for the known option prices (time step j+1).
   Note: a1, b1 and c1 are used to form the RHS vector rhs[] of the tridiagonal system.  */
       sig2 = sigma*sigma;
       t2 = dt/2.0;
       tmp = 1.0-theta_m;  /* 1 - theta (for theta method) */
       for( i=1; i<=ns-2; ++i) { /* Assign elements of the (ns-2)*(ns-2) tridiagonal matrix */
          a[i] = -i*(i*sig2-(r-q))*t2*tmp;
          a1[i] = i*(i*sig2-(r-q))*t2*theta_m;;
          c[i] = -i*(i*sig2+(r-q))*t2*tmp;
          c1[i] = i*(i*sig2+(r-q))*t2*theta_m;;
          b[i] = 1.0+r*dt*tmp+(i*i*sig2)*dt*tmp;
          b1[i]= 1.0-(i*i*sig2+r)*dt*theta_m;
       }
/* Perform LU decomposition of the tridiagonal matrix with:
   diagonal elements contained in the array b[],  upper diagonal elements contained in the_
   array c[]
   and lower diagonal elements in the array a[]. Store the elements of U but not those of L
   (they will be computed from U)
   Matrix U: The diagonal elements of U are stored in the array u[] and the upper diagonal_
   elements of U
   are just c[].
   Matrix L: For the lower triangular matrix L, the diagonal elements are 1 and the lower_
   diagonal elements
   are l[i] = a[i]/u[i-1], where u[] is the upper diagonal of U. */

       u[1] = b[1];
       if (u[1] == 0.0) printf ("ERROR in array u \n");
       for(i=2; i <=ns-2; ++i) {
          u[i] = b[i] - a[i]*c[i-1]/u[i-1];
```

Code excerpt 5.18 (*Continued*).

```
          if (u[i] == 0.0) printf ("ERROR in array u \n");
       }
/*  Set option values at maturity.  Note : opt_vals[0] and opt_vals[ns-1] are the lower and_
    upper
    (put/call) option price boundary values. */
       if (!put) { /* a call */
          for( i=0; i<ns; ++i )
             opt_vals[i] = MAX(s[i]-strike, 0.0 );
       }
       else { /* a put */
          for( i=0; i<ns; ++i)
              opt_vals[i] = MAX(strike - s[i], 0.0);
       }
/* From the option values at maturity (t = nt*dt) calculate values at earlier times (nt-1)*dt_
   etc..   */
      for( j=nt-1; j>=-2; --j) { /* Go two steps past current time (0) so that can evaluate_
      theta */
          time_2mat = T-j*dt;
         for(i=2; i<=ns-3; ++i) /* set up the rhs of equation for Crank-Nicolson method */
             rhs[i] = a1[i]*opt_vals[i-1]+b1[i]*opt_vals[i]+c1[i]*opt_vals[i+1];

/* Incorporate the boundary conditions at the upper/lower asset value boundaries */
         rhs[1] = (a1[1]-a[1])*opt_vals[0]+ b1[1]*opt_vals[1]+c1[1]*opt_vals[2];
         rhs[ns-2] = a1[ns-2]*opt_vals[ns-3]+b1[ns-2]*opt_vals[ns-2]+(c1[ns-2]-c[ns-2])_
         *opt_vals[ns-1];

/* Solve the lower triangular system Ly = b, where y is stored in array work[].
           Compute the elements of L from those of U, l[i] = a[i]/u[i-1].  */
         work[1] = rhs[1];
         for( i=2; i<=ns-2; ++i ) {
           work[i] = rhs[i] - a[i]*work[i-1]/u[i-1];
         }
/* Solve the upper (ns-2)*(ns-2) triangular system Ux = y (where x = opt_vals) */
         opt_vals[ns-2] = work[ns-2]/u[ns-2];
         for( i = ns-2; i >= 1; --i )
            opt_vals[i] = (work[i] - c[i]*opt_vals[i+1])/u[i];
         if (is_american) {  /* take into account early exercise for american options */
            if (put) {  /* a put */
              for(i=1; i<=ns-2; ++i)
                opt_vals[i] = MAX(opt_vals[i],strike-s[i]);
            }
            else { /* a call */
              for(i=1; i<=ns-2; ++i)
                opt_vals[i] = MAX(opt_vals[i],s[i]-strike);
            }
         }
         if (j==0) {
            for (i=0; i < ns; ++i)
                vals[i] = opt_vals[i];
         }
         if ((j==1)||(j==2)||(j==-1)||(j==-2)) { /* Store option values so that can compute_
         theta */
             temp[ind] = opt_vals[n1];
             ++ind;
         }
       }
      if (greeks) {
/* Compute gamma (4th order accuracy) */
        greeks[0] = (-vals[n1+2]+16.0*vals[n1+1]-30.0*vals[n1]+16.0*vals[n1-1]-vals[n1-2])_
        /(12.0*ds*ds);
/* Compute delta (4th order accuracy) */
        greeks[1] = (-vals[n1+2]+8.0*vals[n1+1]-8.0*vals[n1-1]+vals[n1-2])/(12.0*ds);
/* Compute theta (4th order accuracy) */
        greeks[2] = (-temp[0]+8.0*temp[1]-8.0*temp[2]+temp[3])/(12.0*dt);
/* Note: could also compute theta as greeks[2] = (-temp[0]+4.0*temp[1]-3.0*vals[n1])
  /(2.0*dt); */
      }
      *option_value = vals[n1]; /* Return option value */
}
```

Code excerpt 5.18 Function to compute the value of a vanilla option using a uniform grid.

nearest asset grid line. Furthermore, as the number of asset points, n_s, increases the magnitude of d_s will oscillate within the range 0 to $\Delta S/2$.

When $d_s \sim 0$ the grid will be accurate, but when $|d_s| \sim \Delta S/2$ there will be a large pricing error. This gives rise to the oscillating pricing errors shown in Figs. 5.7 and 5.8.

The nonuniform grids are constructed using the techniques mentioned earlier in this section, and also Code excerpt 5.19. We now, irrespective of n_s, arrange for one asset grid line to coincide with the current asset value, S_0, and another asset grid line to coincide with B, the barrier asset price. In Fig 5.6 this corresponds to setting B_L to B and not using B_U.

It can be seen in Figs. 5.7 and 5.8 that in this case the pricing error is very much less, and also does not exhibit the pronounced oscillations that are produced by a uniform grid. In Code excerpt 5.19 we give the computer program which was used to obtained the nonuniform grid values for the down and out call options presented in Figs. 5.7 and 5.8. Although this program only deals with European options it can easily be altered, using the same techniques as in Code excerpt 5.18, to deal with American-style options; this is left as an exercise for the reader.

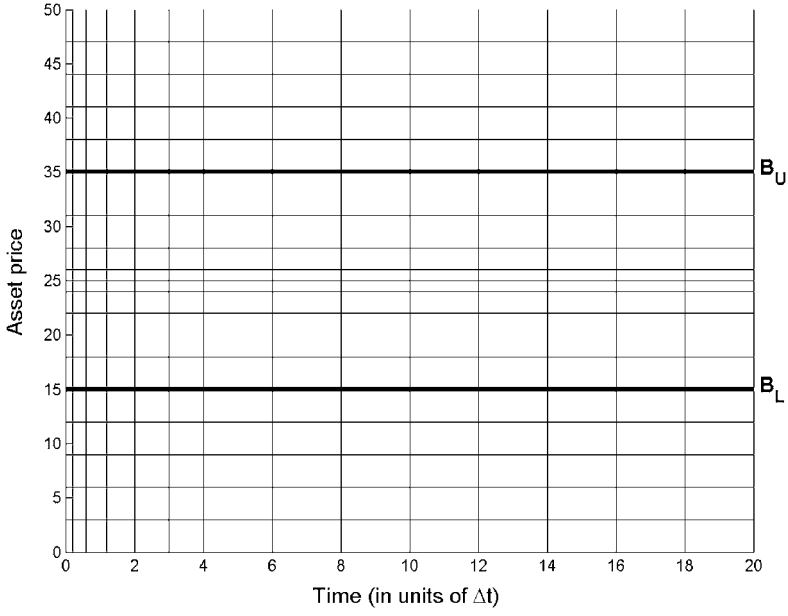

Figure 5.6 A nonuniform grid in which the grid spacing is reduced near current time t, and also in the neighborhood of the asset price 25; this can lead to greater accuracy in the computed option values and the associated *Greeks*. Grid lines are also placed at asset prices of B_{U} and B_{L}; this enables the accurate evaluation of options which have barriers at these asset prices.

```
void barrier_downout(double barrier_level, double theta_m, double asset_price, double sigma,_
        double r, double T,
        double strike, long put, double *option_value, double greeks[], double q, long ns,_
        long nt, double smax, long *ifail)
{
  /* ns - the number of asset intervals
     nt - the number of time intervals
   */
     double *a,*b,*c,*a1,*b1,*c1,*opt_vals,*vals,*rhs,*s,*work,*u;
     double ds,time_step;
     long i,j,barrier_index;
     double tmp,t2,time_2mat,zero = 0.0;
     long n1,n2,ind=0,ns1;
     double sig2,temp[4],ds_plus,ds_minus,temp1,temp2,temp3;
     double D;

      n1 = floor((asset_price/smax)*(double)ns);
      if (n1 < 3) {
         printf ("increase the number of asset points \n");
      }
      n2 = ns - n1;
      ds = asset_price/(double)n1;
      time_step = T/(double)nt;     /* time interval size  */
      ns1 = n1+n2+2; /* number of nodes - including extra grid line*/
      /* allocate the required arrays (all of size ns1+1): a, b, c, a1, b1, c1, opt_vals,_
      vals, rhs, s, work, u */
            .     .     .
      /* set prices below asset_price */
      s[0] = zero;
      s[n1] = asset_price;
      for(i=1; i < n1; ++i )
         s[i] = (double)i * ds;
      /* set prices above asset_price */
      for(i=1; i<= n2+2; ++i ) {
         s[n1+i] = asset_price + (double)i * ds;
      }
/* find out the index corresponding to barrier_level */
      barrier_index = 0;
      while(barrier_level > s[barrier_index]) {
         ++barrier_index;
      }
      if (barrier_level != s[barrier_index]) { /* decrement barrier index */
          --barrier_index;
      }
      if (s[barrier_index] != barrier_level) { /* then barrier does not correspond_
                                                  to an existing grid line  so create another_
                                                  one*/
          for (i=1; i < ns1-barrier_index; ++i) {
              s[barrier_index+1+i] = s[barrier_index] + (double)i*ds;
          }
          ++barrier_index;
          s[barrier_index] = barrier_level;
          if (n1>barrier_index) {
            ++n1;
          }
      }
      /* set up the RHS and LHS coefficients a[], b[] and c[] are the LHS coefficients
         for the unknown option values (time step j) a1[], b1[] and c1[] are the values of the
         RHS coefficients for the known option prices (time step j+1).
         Note: a1, b1 and c1 are used to form the RHS vector rhs[] of the tridiagonal_
         system. */
      sig2 = sigma*sigma;
      t2 = time_step/2.0;
      tmp = 1.0-theta_m;  /* 1 - theta (for theta method) */
      /* assign elements of the (ns1-2)*(ns1-2) tridiagonal matrix */
      for( i=1; i<=ns1-2; ++i) {
          ds_plus  = s[i+1]-s[i];
          ds_minus = s[i] - s[i-1];
          D = ((ds_plus*ds_plus*ds_minus) + (ds_minus*ds_minus*ds_plus));
          temp1 = tmp*time_step/D;
          a[i] = temp1*((r-q)*s[i]*ds_plus*ds_plus) -temp1*ds_plus*(s[i]*s[i]*sig2);
          temp1 = theta_m*time_step/D;
          a1[i] = -(temp1*((r-q)*s[i]*ds_plus*ds_plus) -temp1*ds_plus*(s[i]*s[i]*sig2));
          temp1 = (ds_minus*ds_minus)/D;
          temp2 = ds_minus/D;
```

Code excerpt 5.19.

```
        c[i]  = -time_step*tmp*(temp1*s[i]*(r-q)+(sig2*s[i]*s[i]*temp2));
        c1[i] = time_step*theta_m*(temp1*s[i]*(r-q)+(sig2*s[i]*s[i]*temp2));
        temp1 = ((ds_minus*ds_minus) - (ds_plus*ds_plus))/D;
        temp2 = (ds_minus+ds_plus)/D;
        b[i] = 1.0+time_step*tmp*(r+((r-q)*s[i]*temp1)+(s[i]*s[i]*sig2)*temp2);
        b1[i] = 1.0-time_step*theta_m*(r+((r-q)*s[i]*temp1)+(s[i]*s[i]*sig2)*temp2);
     }
     /* Perform LU decomposition of the tridiagonal matrix with: diagonal elements contained_
        in the array b[],
        upper diagonal elements contained in the array c[]  and lower diagonal elements in_
        the array a[].
        Store the elements of U but not those of L (they will be computed from U)
        Matrix U: The diagonal elements of U are stored in the array u[] and the upper
        diagonal elements of U are just c[].
        Matrix L: For the lower triangular matrix L, the diagonal elements are 1 and the_
        lower diagonal
        elements are l[i] = a[i]/u[i-1], where u[] is the upper diagonal of U. */
     u[1] = b[1];
     if (u[1] == zero) printf ("error in array u \n");
     for( i=2; i <=ns1-2; ++i) {
         u[i] = b[i] - a[i]*c[i-1]/u[i-1];
         if (u[i] == zero) printf ("error in array u \n");
     }
   /* Set option values at maturity. Note : opt_vals[0] and opt_vals[ns1-1] are the lower and_
      upper
      (put/call) option price boundary values. */
     if (!put) { /* a call */
        for( i=0; i<ns1; ++i )
           opt_vals[i] = MAX(s[i]-strike, zero );
        /* now modify option values to include the barrier */
        for( i=0; i <= barrier_index; ++i )
           opt_vals[i] = zero;
    }
     else { /* a put */
        for( i=0; i<ns1; ++i)
             opt_vals[i] = MAX(strike - s[i], zero);
     }
     /* From the option values at maturity, t = nt*time_step, compute
        the values at times (nt-1)*time_step to 0 (current time)
      */
     for( j=nt-1; j>=-2; --j) { /* go two steps past current time so that can evaluate_
     theta */
        time_2mat = T-j*time_step;
        /* set up the rhs of equation for the Theta method */
        for(i=2; i<=ns1-3; ++i)
            rhs[i] = a1[i]*opt_vals[i-1]+b1[i]*opt_vals[i]+c1[i]*opt_vals[i+1];
        /* incorporate the boundary conditions1 at the upper/lower asset value boundaries */
        rhs[1] = (a1[1]-a[1])*opt_vals[0]+ b1[1]*opt_vals[1]+c1[1]*opt_vals[2];
        rhs[ns1-2] = a1[ns1-2]*opt_vals[ns1-3]+b1[ns1-2]*opt_vals[ns1-2]+(c1[ns1-2]-c[ns1-2])_
        *opt_vals[ns1-1];
       /* Solve the lower triangular system Ly = b, where y is stored in array work[].
          Compute the elements of L from those of U, l[i] = a[i]/u[i-1]. */
        work[1] = rhs[1];
        for( i=2; i<=ns1-2; ++i ) {
          work[i] = rhs[i] - a[i]*work[i-1]/u[i-1];
        }
        /* Solve the upper (ns1-2)*(ns1-2) triangular system Ux = y (where x = opt_vals) */
        opt_vals[ns1-2] = work[ns1-2]/u[ns1-2];
        for( i = ns1-2; i >= 1; --i )
           opt_vals[i] = (work[i] - c[i]*opt_vals[i+1])/u[i];
        if (j==0) {
           for (i=0; i < ns1; ++i)
               vals[i] = opt_vals[i];
        }
        /* store option values so that can compute theta */
        if ((j==1)||(j==2)||(j==-1)||(j==-2)) {
            temp[ind] = opt_vals[n1];
            ++ind;
        }
        /* now modify for barrier */
        for( i=0; i <= barrier_index; ++i )
           opt_vals[i] = zero;
     }
    if (greeks) { /* assume an irregular grid */
       ds_minus = s[n1]-s[n1-1];
```

Code excerpt 5.19 (*Continued*).

```
        ds_plus  = s[n1+1]-s[n1];
        D = (ds_minus*ds_minus*ds_plus) + (ds_plus*ds_plus*ds_minus);
        temp1 = ds_minus*ds_minus;
        temp2 = ds_plus*ds_plus;
        temp3 = temp1-temp2;
        /* GAMMA */
       greeks[0] = (ds_minus*vals[n1+1]+ds_plus*vals[n1-1]-vals[n1]*(ds_plus+ds_minus))_
       /(0.5*D);
       /* DELTA */
        greeks[1] = (temp1*vals[n1+1] - temp2*vals[n1-1] - vals[n1]*temp3)/D;
       /*  THETA */
       greeks[2] = (-temp[0]+8.0*temp[1]-8.0*temp[2]+temp[3])/(12.0*time_step);
      /* could also compute theta like this:
         greeks[2] = (-temp[0]+4.0*temp[1]-3.0*vals[n1])/(2.0*time_step); */
     }
      *option_value = vals[n1]; /* Return option value */
    /* deallocate the arrays that were previously allocated */
       .    .    .
}
```

Code excerpt 5.19 Function to compute the value of a European down and out barrier option using a nonuniform grid.

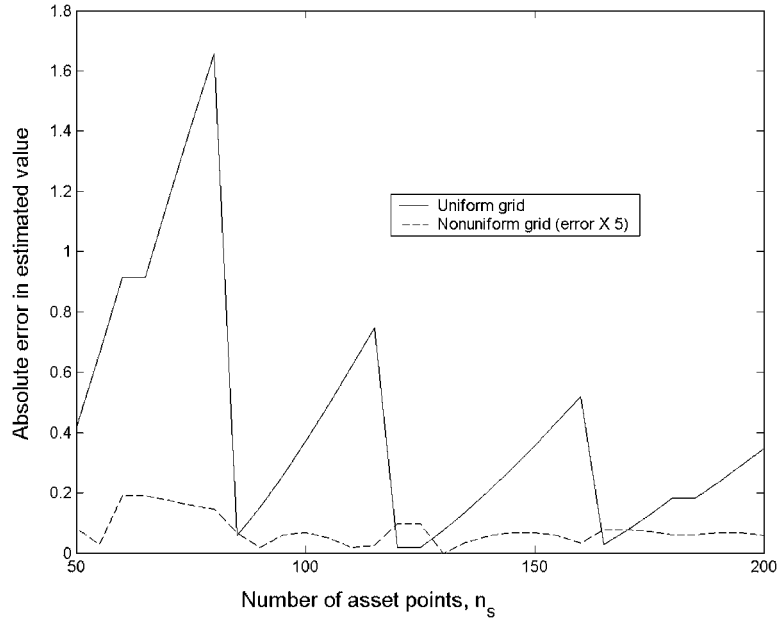

Figure 5.7 The absolute error in the estimated values for a European down and out call barrier option ($B < E$) as the number of asset grid points, n_s, are varied. Here we show a comparison of the results obtained using both uniform and nonuniform grids; logarithmic transformations were not employed. The algorithm for the uniform grid is described in Section 5.4.2, and that for the nonuniform grid is outlined in Section 5.4.3. The Crank–Nicolson method ($\Theta_m = 0.5$) was used and the other parameters were $E = 50.0$, $B = 47.5$, $S_0 = 55.0$, $S_{\max} = 300.0$, $T = 0.5$, $\sigma = 0.2$, $r = \log(1.1)$, $q = 0.0$, $n_t = 100$. The correct option value was 7.6512, which was obtained using the analytic formulae given in Code excerpt 4.7.

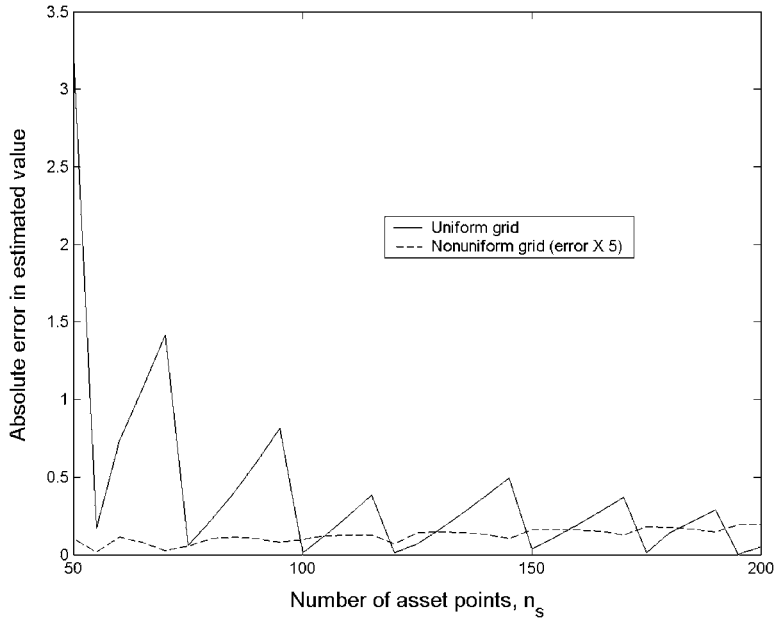

Figure 5.8 The absolute error in the estimated values for a European down and out call barrier option ($E < B$) as the number of asset grid points, n_s, is varied. Here we show a comparison of the results obtained using both uniform and nonuniform grids; logarithmic transformations were not employed. The algorithm for the uniform grid is described in Section 5.4.2 and that for the nonuniform grid is outlined in Section 5.4.3. The Crank–Nicolson method ($\Theta_m = 0.5$) was used and the other parameters were $E = 50.0$, $B = 52.5$, $S_0 = 65.0$, $S_{\max} = 300.0$, $T = 0.5$, $\sigma = 0.2$, $r = \log(1.1)$, $q = 0.0$, $n_t = 100$. The correct option value was 17.0386, which was obtained using the analytic formulae given in Code excerpt 4.7.

5.4.4 The log transformation and uniform grids

Up to this point we have been dealing with the standard Black–Scholes equation, which is:

$$\frac{\partial f}{\partial t} + (r - q)S\frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2} = rf \tag{5.4.47}$$

However, if we introduce the change of variable $Z = \log S$, we obtain the following equation:

$$\frac{\partial f}{\partial t} + b\frac{\partial f}{\partial Z} + \frac{\sigma^2}{2}\frac{\partial^2 f}{\partial Z^2} = rf \tag{5.4.48}$$

where $b = r - q - \sigma^2/2$. This form of the Black–Scholes equation has beneficial numerical properties—see Appendix F.

Derivation of Eq. (5.4.48)

We will now derive an expression for the logarithmic Black–Scholes equation, and show that it agrees with Eq. (5.4.48).

Since $Z = \log S$ we have $\frac{\partial Z}{\partial S} = \frac{1}{S}$. This gives:

$$\frac{\partial f}{\partial S} = \frac{\partial f}{\partial Z}\frac{\partial Z}{\partial S} = \frac{1}{S}\frac{\partial f}{\partial Z}$$

and

$$\frac{\partial^2 f}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{\partial f}{\partial S}\right) = \frac{1}{S^2}\frac{\partial f}{\partial Z} + \frac{1}{S}\frac{\partial}{\partial S}\left(\frac{\partial f}{\partial Z}\right) = -\frac{1}{S^2}\frac{\partial f}{\partial Z} + \frac{1}{S}\frac{\partial Z}{\partial S}\frac{\partial}{\partial Z}\left(\frac{\partial f}{\partial Z}\right)$$

$$\frac{\partial^2 f}{\partial S^2} = -\frac{1}{S^2}\frac{\partial f}{\partial Z} + \frac{1}{S^2}\frac{\partial^2 f}{\partial Z^2}$$

So Eq. (5.4.47) becomes:

$$\frac{\partial f}{\partial t} + \frac{(r-q)S}{S}\frac{\partial f}{\partial Z} - \frac{\sigma^2 S^2}{2S^2}\frac{\partial f}{\partial Z} + \frac{\sigma^2 S^2}{2S^2}\frac{\partial^2 f}{\partial Z^2} = rf$$

thus setting $b = r - q - \sigma^2/2$ we obtain:

$$\frac{\partial f}{\partial t} + b\frac{\partial f}{\partial Z} + \frac{\sigma^2}{2}\frac{\partial^2 f}{\partial Z^2} = rf$$

We will now consider the finite difference discretization of Eq. (5.4.48).

The finite-difference method

Application of the finite-difference method to the log transformed Black–Scholes equation is very similar to that already outlined in Sections 5.4.2 and 5.4.3.

Use of the Θ_m method on Eq. (5.4.48) results in:

$$\frac{f_{i+1,j} - f_{i,j}}{\Delta t} + b\left\{\Theta_m f'_{i+1,j} + \Theta^*_m f'_{i,j}\right\} + \frac{1}{2}\sigma^2\left\{\Theta_m f''_{i+1,j} + \Theta^*_m f''_{i,j}\right\}$$
$$= r\left\{\Theta_m f_{i+1,j} + \Theta^*_m f_{i,j}\right\}$$

where $\Theta^*_m = 1 - \Theta_m$. Applying a uniform discretization at node (i, j) we obtain:

$$f_{i+1,j} - f_{i,j} + \frac{b\Delta t\mathcal{A}_1}{2\Delta Z} + \frac{\sigma^2\Delta t\mathcal{A}_2}{2\Delta Z^2} = r\Delta t\left\{\Theta_m f_{i+1,j} + \Theta^*_m f_{i,j}\right\} \qquad (5.4.49)$$

where

$$\mathcal{A}_1 = \Theta_m\{f_{i+1,j+1} - f_{i+1,j-1}\} + \Theta^*_m\{f_{i,j+1} - f_{i,j-1}\}$$
$$\mathcal{A}_2 = \Theta_m\{f_{i+1,j+1} - 2f_{i+1,j} + f_{i+1,j-1}\} + \Theta^*_m\{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}\}$$

Collecting like terms we obtain:

$$\mathcal{B}_1 f_{i,j-1} + \mathcal{B}_2 f_{i,j} + \mathcal{B}_3 f_{i,j+1} + \mathcal{C}_1 f_{i+1,j-1} + \mathcal{C}_2 f_{i+1,j} + \mathcal{C}_3 f_{i+1,j+1} = 0$$

where

$$\mathcal{B}_1 = \frac{-\Theta_m^* b\Delta t}{2\Delta Z} + \frac{\Theta_m^* \sigma^2 \Delta t}{2\Delta Z^2}$$

$$\mathcal{B}_2 = -1 - r\Delta t\Theta_m^* - \frac{\Theta_m^* \sigma^2 \Delta t}{\Delta Z^2}$$

$$\mathcal{B}_3 = \frac{\Theta_m^* b\Delta t}{2\Delta Z} + \frac{\Theta_m^* \sigma^2 \Delta t}{2\Delta Z^2}$$

$$\mathcal{C}_1 = \frac{\Theta_m \sigma^2 \Delta t}{2\Delta Z^2} - \frac{\Theta_m b\Delta t}{2\Delta Z}$$

$$\mathcal{C}_2 = 1 - r\Delta t\Theta_m - \frac{\Theta_m \sigma^2 \Delta t}{\Delta Z^2}$$

$$\mathcal{C}_3 = \frac{\Theta_m b\Delta t}{2\Delta Z} + \frac{\Theta_m \sigma^2 \Delta t}{2\Delta Z^2}$$

If we rearrange we have the following equation:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1}$$

where:

$$a_j = \frac{(1-\Theta_m)\Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\} \tag{5.4.50}$$

$$b_j = 1 + (1-\Theta_m)\Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\} \tag{5.4.51}$$

$$c_j = -\frac{(1-\Theta_m)\Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\} \tag{5.4.52}$$

$$\bar{a}_j = -\frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\} \tag{5.4.53}$$

$$\bar{b}_j = 1 - \Theta_m \Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\} \tag{5.4.54}$$

$$\bar{c}_j = \frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\} \tag{5.4.55}$$

It can be seen that, unlike in Section 5.4.2, the coefficients in Eqs. (5.4.50)–(5.4.55) are independent of the asset price index j.

When $\Theta_m = 0.5$ (the Crank–Nicolson method) we have the following coefficients:

$$a_j = -\bar{a}_j = \frac{\Delta t}{4\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\}$$

$$b_j = 1 + \frac{\Delta t}{2}\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\}$$

$$c_j = -\bar{c}_j = -\frac{\Delta t}{4\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\}$$

$$\bar{b}_j = 1 - \frac{\Delta t}{2}\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\}$$

Table 5.7 Valuation results and pricing errors for a vanilla American put option using a uniform grid with and without a logarithmic transformation; the implicit method and Crank–Nicolson method are used

Time	Value	$\Theta_m = 0.0$		$\Theta_m = 0.5$	
		BS	Log *BS*	*BS*	Log *BS*
0.1	0.7599	1.4733×10^{-2}	7.7803×10^{-3}	1.4719×10^{-2}	7.6716×10^{-3}
0.2	0.8335	4.5838×10^{-2}	1.2924×10^{-2}	4.5682×10^{-2}	1.1997×10^{-2}
0.3	0.8921	6.4218×10^{-2}	1.4125×10^{-2}	6.3800×10^{-2}	1.2567×10^{-2}
0.4	0.9403	7.4699×10^{-2}	1.6559×10^{-2}	7.3924×10^{-2}	1.4655×10^{-2}
0.5	0.9812	8.0297×10^{-2}	1.8471×10^{-2}	7.9101×10^{-2}	1.6041×10^{-2}
0.6	1.0167	8.2796×10^{-2}	1.9125×10^{-2}	8.1135×10^{-2}	1.6067×10^{-2}
0.7	1.0479	8.3285×10^{-2}	1.8959×10^{-2}	8.1131×10^{-2}	1.5273×10^{-2}
0.8	1.0758	8.2470×10^{-2}	1.8408×10^{-2}	7.9803×10^{-2}	1.4159×10^{-2}
0.9	1.1009	8.0829×10^{-2}	1.7756×10^{-2}	7.7647×10^{-2}	1.3020×10^{-2}
1.0	1.1237	7.8646×10^{-2}	1.7138×10^{-2}	7.4947×10^{-2}	1.1997×10^{-2}
1.1	1.1445	7.6164×10^{-2}	1.6643×10^{-2}	7.1961×10^{-2}	1.1174×10^{-2}
1.2	1.1637	7.3514×10^{-2}	1.6290×10^{-2}	6.8803×10^{-2}	1.0552×10^{-2}
1.3	1.1813	7.0785×10^{-2}	1.6092×10^{-2}	6.5594×10^{-2}	1.0143×10^{-2}
1.4	1.1977	6.8080×10^{-2}	1.6042×10^{-2}	6.2419×10^{-2}	9.9309×10^{-3}
1.5	1.2129	6.5424×10^{-2}	1.6128×10^{-2}	5.9295×10^{-2}	9.8909×10^{-3}

The accurate values (obtained using a logarithmic transformed grid with $n_s = 1000$ and $n_t = 1000$) are presented in the column labelled "Value". The absolute pricing errors, *ABS* (*accurate value* − *estimated value*) presented in the column labelled *BS* were obtained using a standard uniform grid (as outlined in Section 5.4.2), and those in the column labelled Log *BS* use a uniform grid and logarithmic transformation as explained in this section. The maturity of the option was varied from 0.1 years to 1.5 years, and the other parameters were: $S = 9.0$, $X = 9.7$, $r = 0.1$, $q = 0.0$, $\sigma = 0.30$, $S_{\max} = 100.0$, $n_s = 50$, and $n_t = 50$.

The method of using the finite-difference grid to compute option prices is identical to that already outlined in Section 5.4.2, which solves the standard (non-logarithmic) Black–Scholes equation. Table 5.7 compares the results obtained with and without a logarithmic transformation.

5.4.5 The log transformation and nonuniform grids

In the previous section we considered the use of a uniform grid to discretize the logarithmically transformed Black–Scholes equation:

$$\frac{\partial f}{\partial t} + b\frac{\partial f}{\partial Z} + \frac{\sigma^2}{2}\frac{\partial^2 f}{\partial Z^2} = rf \tag{5.4.56}$$

where

$$b = r - q - \frac{\sigma^2}{2} \quad \text{and} \quad Z = \log S$$

Here we will generalize these results and use a nonuniform grid to solve Eq. (5.4.56).

Our description will be very brief since most of the details have already been discussed in previous sections. Here we are only concerned with the finite-difference approximation and derive the equations that need to be solved at each time step. Later, in Section 5.4.6, we will apply our results to solving a European double knockout barrier option.

The finite-difference approximation

At the grid node (i, j) we have

$$\Delta Z_j^- = Z_j - Z_{j-1} \quad \text{and} \quad \Delta Z_j^+ = Z_j + 1 - Z_j$$

Following Section 5.4.2 the first and second derivatives of f w.r.t. Z are

$$f''_{i+1,j} = \frac{2\{\Delta Z_j^+ f_{i+1,j-1} + \Delta Z_j^- f_{i+1,j+1} - \Delta Z_j^- f_{i+1,j} - \Delta Z_j^+ f_{i+1,j}\}}{(\Delta Z_j^+)^2 \Delta Z_j^- + (\Delta Z_j^-)^2 \Delta Z_j^+}$$

and

$$f'_{i+1,j} = \frac{(\Delta Z_j^-)^2 f_{i+1,j+1} + ((\Delta Z^+)^2 - (\Delta Z_j^-)^2) f_{i+1,j} - (\Delta Z_j^+)^2 f_{i+1,j-1}}{(\Delta Z_j^+)^2 \Delta Z_j^- + (\Delta Z_j^-)^2 \Delta Z^+}$$

Then discretizing Eq. (5.4.56) in the *usual manner* we obtain

$$\begin{aligned} &\frac{f_{i+1,j} - f_{i,j}}{\Delta t} + b\left\{\Theta_m f'_{i+1,j} + \Theta_m^* f'_{i,j}\right\} + \frac{\sigma^2}{2}\left\{\Theta_m f''_{i+1,j} + \Theta_m^* f''_{i,j}\right\} \\ &\quad = r\left\{\Theta_m f_{i+1,j} + \Theta_m^* f_{i,j}\right\} \end{aligned}$$

where $\Theta_m^* = 1 - \Theta_m$. Letting $D = (\Delta Z_j^+)^2 \Delta Z_j^- + (\Delta Z_j^-)^2 \Delta Z_j^+$ we obtain

$$\begin{aligned} &r\Delta t\left(\Theta_m f_{i+1,j} + \Theta_m^* f_{i,j}\right) \\ &\quad = f_{i+1,j} - f_{i,j} + \frac{b\Delta t \mathcal{A}_1}{D} + \frac{\sigma^2 \Delta t \mathcal{A}_2}{D} \end{aligned} \tag{5.4.57}$$

where

$$\begin{aligned} \mathcal{A}_1 = \Theta_m\big[&f_{i+1,j+1}\left(\Delta Z_j^-\right)^2 - f_{i+1,j-1}\left(\Delta Z_j^+\right)^2 \\ &- f_{i+1,j}\left\{\left(\Delta Z_j^-\right)^2 - \left(\Delta Z_j^+\right)^2\right\}\big] \\ + \Theta_m^*\big[&f_{i,j+1}\left(\Delta Z_j^-\right)^2 - f_{i,j-1}\left(\Delta Z_j^+\right)^2 - f_{i,j}\left\{\left(\Delta Z_j^-\right)^2 - \left(\Delta Z_j^+\right)^2\right\}\big] \\ \mathcal{A}_2 = \Theta_m\big[&f_{i+1,j+1}\Delta Z_j^- + f_{i+1,j-1}\Delta Z_j^+ - f_{i+1,j}\left\{\Delta Z_j^- + \Delta Z_j^+\right\}\big] \\ + \Theta_m^*\big[&f_{i,j+1}\Delta Z_j^- + f_{i,j-1}\Delta Z_j^+ - f_{i,j}\left\{\Delta Z_j^- + \Delta Z_j^+\right\}\big] \end{aligned}$$

Collecting like terms we obtain:

$$\mathcal{B}_1 f_{i,j-1} + \mathcal{B}_2 f_{i,j} + \mathcal{B}_3 f_{i,j+1} + \mathcal{C}_1 f_{i+1,j-1} + \mathcal{C}_2 f_{i+1,j} + \mathcal{C}_3 f_{i+1,j+1} = 0$$

where

$$\mathcal{B}_1 = \frac{-\Theta_m^* b \Delta t (\Delta Z_j^+)^2}{D} + \frac{\Theta_m^* \sigma^2 \Delta t \Delta Z_j^+}{D}$$

$$\mathcal{B}_2 = -1 - r\Delta t \Theta_m^* - \frac{\Theta_m^* \sigma^2 \Delta t (\Delta Z_j^- + \Delta Z_j^+)}{D} - \frac{\Theta_m^* b \Delta t \{(\Delta Z_j^-)^2 - (\Delta Z_j^+)^2\}}{D}$$

$$\mathcal{B}_3 = \frac{\Theta_m^* b \Delta t (\Delta Z_j^-)^2}{D} + \frac{\Theta_m^* \sigma^2 \Delta t \Delta Z^-}{D}$$

$$\mathcal{C}_1 = \frac{\Theta_m \sigma^2 \Delta t \Delta Z_j^+}{D} - \frac{\Theta_m b \Delta t (\Delta Z_j^+)^2}{D}$$

$$\mathcal{C}_2 = 1 - r\Delta t \Theta_m - \frac{\Theta_m b \Delta t \{(\Delta Z_j^-)^2 - (\Delta Z_j^+)^2\}}{D} - \frac{\Theta_m \sigma^2 \Delta t \{\Delta Z_j^- + \Delta Z_j^+\}}{D}$$

$$\mathcal{C}_3 = \frac{\Theta_m b \Delta t (\Delta Z_j^-)^2}{D} + \frac{\Theta_m \sigma^2 \Delta t \Delta Z_j^-}{D}$$

If we rearrange we have the following equation:

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j = \bar{a}_j f_{i+1,j-1} + \bar{b}_j f_{i+1,j} + \bar{c}_j f_{i+1,j+1} \tag{5.4.58}$$

where:

$$a_j = (1 - \Theta_m)\Delta t \left\{ \frac{b(\Delta Z_j^+)^2}{D} - \frac{\sigma^2 \Delta Z_j^+}{D} \right\} \tag{5.4.59}$$

$$b_j = 1 + \Delta t (1 - \Theta_m) \times \left\{ r - \frac{\sigma^2 (\Delta Z_j^- + \Delta Z_j^+)}{D} - \frac{b\{(\Delta Z_j^-)^2 - (\Delta Z_j^+)^2\}}{D} \right\} \tag{5.4.60}$$

$$c_j = (1 - \Theta_m)\Delta t \left\{ \frac{-b(\Delta Z_j^-)^2}{D} - \frac{\sigma^2 \Delta Z_j^-}{D} \right\} \tag{5.4.61}$$

$$\bar{a}_j = \Theta_m \Delta t \left\{ \frac{\sigma^2 \Delta Z_j^+}{D} - \frac{b(\Delta Z_j^+)^2}{D} \right\} \tag{5.4.62}$$

$$\bar{b}_j = 1 - \Theta_m r \Delta t - \Theta_m \Delta t \left\{ \frac{b\{(\Delta Z_j^-)^2 - (\Delta Z_j^+)^2\}}{D} + \frac{\sigma^2 \{\Delta Z_j^- + \Delta Z_j^+\}}{D} \right\} \tag{5.4.63}$$

$$\bar{c}_j = \Theta_m \Delta t \left\{ \frac{b(\Delta Z_j^-)^2}{D} + \frac{\sigma^2 \Delta Z_j^-}{D} \right\} \tag{5.4.64}$$

The incorporation of boundary conditions and the solution of Eq. (5.4.58) is similar in manner to that already discussed in Section 5.4.2. If further details

are required Code excerpt 5.19, which uses a nonuniform grid to solve the log transformed Black–Scholes equation, can be consulted.

When a uniform grid is used $\Delta Z_j^+ = \Delta Z_j^- = \Delta Z$ and therefore

$$D = \left(\Delta Z_j^+\right)^2 \Delta Z_j^- + \left(\Delta Z_j^-\right)^2 \Delta Z_j^+ = 2(\Delta Z)^3$$

$$\frac{(\Delta Z_j^+)^2}{D} = \frac{(\Delta Z_j^-)^2}{D} = \frac{(\Delta Z)^2}{2(\Delta Z)^3} = \frac{1}{2\Delta Z}$$

$$\frac{\Delta Z_j^+}{D} = \frac{\Delta Z_j^-}{D} = \frac{1}{2\Delta Z^2} \quad \text{and} \quad \frac{(\Delta Z_j^+)^2 - (\Delta Z_j^-)^2}{D} = 0$$

In these circumstances

$$a_j = \frac{(1-\Theta_m)\Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\}$$

$$b_j = 1 + \Delta t(1-\Theta_m)\left\{r - \frac{\sigma^2}{\Delta Z^2}\right\}$$

$$c_j = (1-\Theta_m)\Delta t\left\{\frac{-b}{2\Delta Z} - \frac{\sigma^2}{2\Delta Z^2}\right\}$$

$$\bar{a}_j = -\frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\}$$

$$\bar{b}_j = 1 - \Theta_m \Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\}$$

$$\bar{c}_j = \frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\}$$

which are the same as Eqs. (5.4.50)–(5.4.55) in Section 5.4.4.

5.4.6 The double knockout call option

The purpose of this section is to provide an example that illustrates the benefits to be gained from using both the log transformed Black–Scholes equation and also a nonuniform grid.

The problem we will consider is the European double knockout call option with strike price E, and expiry date T. This is a barrier option with both an upper barrier at B_U and a lower barrier at B_L. If, during the life of the option, the asset price either goes above the upper barrier or below the lower barrier, then the option becomes worthless. If, on the other hand, the asset price stays between the barriers then the option has value $\max(S_T - E, 0)$, where S_T is the asset price at time T.

This problem has been previously investigated by Boyle and Tian (1998), henceforth referred to as BT, who used an explicit finite-difference method based on a modified trinomial lattice. The method we use here is based on the finite-difference equations given in Section 5.4.5, and all the results obtained by using the function `dko_call` (see Code excerpt 5.20) are presented in Tables 5.8–5.12.

```
void dko_call(double lower_barrier, double upper_barrier, double theta_m,
      double S0, double sigma_array[], double sigma_times[], long n_sigma, double r,
      double opt_mat, double X, double *option_value, double greeks[], double q,
      long ns_below_S0, long ns_above_S0, long nt, long *iflag)
{
/* Input parameters:
   =================
   lower_barrier         - the asset price corresponding to the lower barrier,
   upper_barrier         - the asset price corresponding to the upper barrier,
   theta_m               - the value of theta used for the finite difference method,
   S0                    - the current price of the underlying asset,
   sigma_array[]         - an array containing values of the volatility: sigma_array[0] is the_
                           first value of the volatility,
                           sigma_array[1] is the second value of the volatility, etc..,
   sigma_times[]         - an array containing the times for different volatilities:_
                           sigma_times[0] is the time corresponding to
                           the first volatility, sigma_times[1] is the time corresponding to_
                           the second volatility, etc..,
   n_sigma               - the number of elements in sigma_array[], and sigma_times[],
   r                     - the interest rate,
   opt_mat               - the time to maturity,
   X                     - the strike price,
   q                     - the continuous dividend yield,
   ns_below_S0           - the number of asset intervals below the current price S0,
   ns_above_S0           - the number of asset intervals above the current price S0,
   nt                    - the number of time intervals.
   Output parameters:
   ==================
   option_value          - the value of the option,
   greeks[]              - the hedge statistics output as follows: greeks[0] is gamma,_
                           greeks[1] is delta, and greeks[2] is theta,
   iflag                 - an error indicator.
*/
     double *a,*b,*c,*vals,*a1,*b1,*c1,*opt_vals,*rhs,*z,*delta,*gamma,*work,*u;
     double dt,dz,dz1,dz2,zmax,zmin;
     long i,j;
     double tmp,t2,t4,dt2;
     long ind=0,n1,n2,ns1;
     double ds,log_asset,sig2,alpha,v2,b_fac,temp[4];
     double zero = 0.0;
     long barrier_index,ind2;
     double dz_shift,time_step,log_barrier_level1,log_barrier_level2;
     double temp1, temp2, ds_plus, ds_minus, bb, D;
     double curr_time;

      if (S0 >= upper_barrier) printf ("ERROR current asset price is greater than_
      upper_barrier \n");
      if (lower_barrier >= S0) printf("ERROR lower barrier is greater than current asset_
      price \n");
      if (S0 <= zero) printf ("ERROR asset price is not > 0 \n");
      if (upper_barrier <= lower_barrier) printf ("ERROR upper_barrier must be >_
      lower_barrier \n");
      log_asset = log(S0);
      log_barrier_level1 = log(lower_barrier);
      log_barrier_level2 = log(upper_barrier);
      dz1 = (log_asset-log_barrier_level1)/(double)ns_below_S0;
      n1 = ns_below_S0;
/* Include 5 extra points above the asset price so that don't get discontinuity in grid_
   spacing
   which may adversely affect the computation of the greeks */
      n2 = ns_above_S0 + 5;
      dz_shift = dz1*5.0; /* shift caused by extra 5 grid points */
      dz2 = (log_barrier_level2-log_asset-dz_shift)/(double)ns_above_S0;
      dt = opt_mat/(double)nt;     /* time interval size  */
      time_step = dt;
      --n2;
      ns1 = n1+n2+2;
/* Set up the RHS and LHS coefficients a[], b[] and c[] are the LHS coefficients for the_
   unknown option
   values (time step j) a1[], b1[] and c1[] are the values of the RHS coefficients for the_
   known option prices
   (time step j+1). Note: a1, b1 and c1 are used to form the RHS vector rhs[] of the_
   tridiagonal system. */
/* Allocate the required arrays (all of size (ns1+2): a,b,c,a1,b1,c1,opt_vals,vals,_
   rhs,z,delta,gamma,work,u */
```

Code excerpt 5.20.

```
            .   .   .
/* Set up the RHS and LHS coefficients a[], b[] and c[] are the LHS coefficients
   for the unknown option values (time step j) a1[], b1[] and c1[] are the values of the
   RHS coefficients for the known option prices (time step j+1). Note: a1, b1 and c1 are used_
   to form the RHS
   vector rhs[] of the tridiagonal system. */
/* Set grid line asset values, set one grid spacing to align with the asset price, then won't_
   have to
   interpolate to get the option value */
      z[n1] = log_asset;
      for (i = 1; i <=n1; ++i) /* This should be the fine mesh */
         z[n1-i] = log_asset - (double)i*dz1;
      for (i = 1; i <= 5; ++i) /* Include 5 extra fine mesh points here */
         z[n1+i] = log_asset + (double)i*dz1;
      for (i = 6; i <= n2+2; ++i) { /* The coarse mesh */
         j = i - 5;
         z[n1+i] = z[n1+5] + (double)j*dz2;
      }
/*  Set option values at maturity (for a call). Note : opt_vals[0] and opt_vals[ns1-1] are the
    lower and upper
    (put/call) option price boundary values. */
      for( i=1; i<ns1; ++i ) {
         opt_vals[i] = MAX(exp(z[i])-X, zero);
      }
      opt_vals[0] = zero;
      opt_vals[ns1-1] = zero;
      tmp = 1.0-theta_m;  /* 1 - theta (for theta method) */
      curr_time = -1.0;
      ind2 = n_sigma - 1;
      for( j=nt-1; j>=-2; --j) { /* Iterate from maturity to current time  */
          if ((ind2 >= 0) && (curr_time <= sigma_times[ind2])) {
              sig2 = sigma_array[ind2]*sigma_array[ind2];
              t2 = time_step/2.0;
              bb = r - q - (sig2/2.0);
              --ind2;
              for( i=1; i<=ns1-2; ++i) { /* Assign elements of the (ns1-2)*(ns1-2) tridiagonal_
              matrix */
                ds_plus  = z[i+1]-z[i];
                ds_minus = z[i] - z[i-1];
                D = ((ds_plus*ds_plus*ds_minus) + (ds_minus*ds_minus*ds_plus));
                temp1 = tmp*time_step/D;
                a[i] = temp1*(bb*ds_plus*ds_plus) -temp1*ds_plus*(sig2);
                temp1 = theta_m*time_step/D;
                a1[i] = temp1*ds_plus*(sig2)-temp1*(bb*ds_plus*ds_plus);
                temp1 = (ds_minus*ds_minus)/D;
                temp2 = ds_minus/D;
                c[i]  = -time_step*tmp*(temp1*bb+(sig2*temp2));
                c1[i] = time_step*theta_m*(temp1*bb+(sig2*temp2));
                temp1 = ((ds_minus*ds_minus) - (ds_plus*ds_plus))/D;
                temp2 = (ds_minus+ds_plus)/D;
                b[i] = 1.0+time_step*tmp*(r+(bb*temp1)+(sig2)*temp2);
                b1[i] = 1.0-time_step*theta_m*(r+(bb*temp1)+(sig2)*temp2);
            }
            u[1] = b[1];
            if (u[1] == zero) printf ("ERROR in array u \n");
              for( i=2; i <=ns1-2; ++i) {
                 u[i] = b[i] - a[i]*c[i-1]/u[i-1];
                 if (u[i] == zero) printf ("ERROR in array u \n");
              }
          }
          curr_time = j*dt;
/* Set up the rhs of equation for the theta method */
          for(i=2; i<=ns1-3; ++i)
              rhs[i] = a1[i]*opt_vals[i-1]+b1[i]*opt_vals[i]+c1[i]*opt_vals[i+1];
/* Incorporate the boundary conditions1 at the upper/lower asset value boundaries */
          rhs[1] = (a1[1]-a[1])*opt_vals[0]+ b1[1]*opt_vals[1]+c1[1]*opt_vals[2];
          rhs[ns1-2] = a1[ns1-2]*opt_vals[ns1-3]+b1[ns1-2]*opt_vals[ns1-2]+(c1[ns1-2]-c[ns1-2])_
          *opt_vals[ns1-1];
/* Solve the lower triangular system Ly = b, where y is stored in array work[]. Compute the_
   elements of L from those of U, l[i] = a[i]/u[i-1]. */
          work[1] = rhs[1];
          for( i=2; i<=ns1-2; ++i ) {
            work[i] = rhs[i] - a[i]*work[i-1]/u[i-1];
          }
```

Code excerpt 5.20 (*Continued*).

```
/* Solve the upper (ns1-2)*(ns1-2) triangular system Ux = y (where x = vold) */
        opt_vals[ns1-2] = work[ns1-2]/u[ns1-2];
        for( i = ns1-2; i >= 1; --i )
           opt_vals[i] = (work[i] - c[i]*opt_vals[i+1])/u[i];
        if (j==0) {
           for (i=0; i < ns1; ++i)
               vals[i] = opt_vals[i];
        }
/* Store option values so that can compute theta */
        if ((j==1)||(j==2)||(j==-1)||(j==-2)) {
            temp[ind] = opt_vals[n1];
            ++ind;
        }
      }
     if (greeks) {
/* Compute gamma and delta (4th order accuracy) */
       greeks[1] = (-vals[n1+2]+8.0*vals[n1+1]-8.0*vals[n1-1]+vals[n1-2])/(12.0*dz1);
/* Compute gamma (4th order accuracy) - use chain rule to obtain derivative wrt S */
       greeks[0] = (-vals[n1+2]+16.0*vals[n1+1]-30.0*vals[n1]+16.0*vals[n1-1]-vals[n1-2])_
       /(12.0*dz1*dz1);
       greeks[0] = greeks[0]-greeks[1];
       greeks[0] = greeks[0]/(S0*S0);
       greeks[1] = greeks[1]/S0;
/* Compute theta (4th order accuracy) */
       greeks[2] = (-temp[0]+8.0*temp[1]-8.0*temp[2]+temp[3])/(12.0*dt);
       /* could also compute theta as: greeks[2] = (-temp[0]+4.0*temp[1]-3.0*vals[n1])_
       /(2.0*dt);  */
     }
     *option_value = vals[n1];
}
```

Code excerpt 5.20 Code excerpt 5.20 Function to compute the value and Greeks of a European double knockout call option using a nonuniform grid and a logarithmic transformation.

Table 5.8 Estimated value of a European double knockout call option

Time steps (n)	Estimated value	Boyle and Tian (1998)
50	1.4569	1.4238
100	1.4578	1.4437
200	1.4583	1.4495
300	1.4583	1.4524
400	1.4584	1.4542
500	1.4584	1.4553
600	1.4584	1.4557
700	1.4584	1.4559
800	1.4584	1.4563
900	1.4584	1.4565
1000	1.4584	1.4566
2000	1.4584	1.4576
3000	1.4584	1.4578
4000	1.4584	1.4580
5000	1.4584	1.4581

The values in column two were computed by the function `dko_call`, and those in column three are the results reported in Table 2 of Boyle and Tian (1998). The model parameters were: current asset price $S = 95.0$, exercise price $E = 100.0$, volatility $\sigma = 0.25$, maturity $\tau = 1.0$, interest rate $r = 0.1$, dividend yield $q = 0.0$. The upper barrier level is set at 140.0 and the lower barrier is set at 90.0. The other parameters used by the function `dko_call` were: `nt` $= n$, `ns_below_S0` $= n/2$, `ns_above_S0` $= n/2$, and $\Theta_m = 0.5$ (i.e., the Crank–Nicolson method).

Table 5.9 The estimated values of European down and out call options calculated by the function `dko_call`

Time steps	Stock price					
	92	91	90.5	90.4	90.3	90.2
50	2.5652	1.3046	0.6588	0.5282	0.3971	0.2653
100	2.5221	1.2816	0.6466	0.5182	0.3894	0.2601
200	2.5104	1.2758	0.6435	0.5157	0.3875	0.2588
300	2.5080	1.2747	0.6429	0.5152	0.3871	0.2585
400	2.5072	1.2743	0.6427	0.5150	0.3869	0.2584
500	2.5069	1.2742	0.6426	0.5149	0.3869	0.2584
600	2.5067	1.2741	0.6425	0.5149	0.3868	0.2583
700	2.5066	1.2740	0.6425	0.5149	0.3868	0.2583
800	2.5065	1.2740	0.6424	0.5148	0.3868	0.2583
900	2.5065	1.2739	0.6424	0.5148	0.3868	0.2583
1000	2.5064	1.2739	0.6424	0.5148	0.3868	0.2583
2000	2.5063	1.2738	0.6424	0.5148	0.3868	0.2583
Closed form	2.5063	1.2738	0.6424	0.5148	0.3868	0.2583

The fixed model parameters were: exercise price $E = 100.0$, volatility $\sigma = 0.25$, maturity $\tau = 1.0$, interest rate $r = 0.1$, dividend yield $q = 0.0$, and the lower barrier is set at 90.0. The other parameters used by the function `dko_call` were: `nt` $= n$, `ns_below_S0` $= n/2$, `ns_above_S0` $= n/2$, `upper_barrier` $= 1000.0$, `lower_barrier` $= 90.0$, and $\Theta_m = 0.5$ (i.e., the Crank–Nicolson method).

Table 5.10 The estimated values of European down and out call options as calculated by the function `dko_call`

Time steps	Stock price					
	92	91	90.5	90.4	90.3	90.2
50	2.5572	1.3005	0.6567	0.5266	0.3958	0.2645
100	2.5181	1.2796	0.6455	0.5174	0.3888	0.2597
200	2.5084	1.2748	0.6429	0.5153	0.3872	0.2586
300	2.5067	1.2741	0.6425	0.5149	0.3869	0.2584
400	2.5062	1.2738	0.6424	0.5148	0.3868	0.2583
500	2.5061	1.2738	0.6424	0.5148	0.3868	0.2583
600	2.5061	1.2737	0.6423	0.5148	0.3867	0.2583
700	2.5060	1.2737	0.6423	0.5147	0.3867	0.2583
800	2.5060	1.2747	0.6423	0.5147	0.3867	0.2583
900	2.5060	1.2737	0.6423	0.5147	0.3867	0.2583
1000	2.5060	1.2737	0.6423	0.5147	0.3867	0.2583
2000	2.5061	1.2737	0.6423	0.5147	0.3867	0.2583
Closed form	2.5063	1.2738	0.6424	0.5148	0.3868	0.2583

The fixed parameters used were: exercise price $E = 100.0$, volatility $\sigma = 0.25$, maturity $\tau = 1.0$, interest rate $r = 0.1$, dividend yield $q = 0.0$, and the lower barrier is set at 90.0. The other parameters used by the function `dko_call` were: `nt` $= n$, `ns_below_S0` $= n/2$, `ns_above_S0` $= n/2$, `upper_barrier` $= 1000.0$, `lower_barrier` $= 90.0$, and $\Theta_m = 0.0$ (i.e., the implicit method).

Table 5.11 The estimated values of European double knockout call options computed by the function `dko_call`

Time steps	Stock price					
	92	91	90.5	90.4	90.3	90.2
50	0.6251 (0.6184)	0.3189 (0.3177)	0.1610	0.1290	0.0969	0.0647
100	0.6260 (0.6212)	0.3194 (0.3184)	0.1613	0.1292	0.0971	0.0649
200	0.6263 (0.6228)	0.3196 (0.3186)	0.1613	0.1293	0.0972	0.0649
300	0.6263 (0.6236)	0.3196 (0.3187)	0.1613	0.1293	0.0972	0.0649
400	0.6263 (0.6242)	0.3196 (0.3189)	0.1613	0.1293	0.0972	0.0649
500	0.6263 (0.6252)	0.3196 (0.3190)	0.1613	0.1293	0.0972	0.0649
600	0.6263 (0.6253)	0.3196 (0.3191)	0.1613	0.1293	0.0972	0.0649
700	0.6263 (0.6253)	0.3196 (0.3191)	0.1613	0.1293	0.0972	0.0649
800	0.6263 (0.6255)	0.3196 (0.3192)	0.1613	0.1293	0.0972	0.0649
900	0.6263 (0.6256)	0.3196 (0.3192)	0.1613	0.1293	0.0972	0.0649
1000	0.6263 (0.6255)	0.3196 (0.3192)	0.1613	0.1293	0.0972	0.0649
2000	0.6263 (0.6260)	0.3196 (0.3195)	0.1613	0.1293	0.0972	0.0649

In column 2 and column 3 the values given in Boyle and Tian (1998), Table 5, are shown for comparison. The fixed model parameters were: exercise price $E = 100.0$, volatility $\sigma = 0.25$, dividend yield $q = 0.0$, maturity $\tau = 1.0$, interest rate $r = 0.1$, the lower barrier is set at 90.0, and the upper barrier is set at 140.0. The other parameters used by the function `dko_call` were: `nt` $= n$, `ns_below_S0` $= n/2$, `ns_above_S0` $= n/2$, and $\Theta_m = 0.5$ (i.e., the Crank–Nicolson method).

Table 5.12 The estimated Greeks for European double knockout call options computed by the function `dko_call`

Asset price	Gamma	Delta	Theta
95.0	−0.0165 (−0.0166)	0.2536 (0.2551)	2.3982 (2.3928)
92.0	−0.0141 (−0.0141)	0.2998 (0.3016)	1.0268 (1.0242)
91.0	−0.0129 (−0.0130)	0.3133 (0.3151)	0.5237 (0.5224)
90.5	−0.0123 (−0.0123)	0.3196 (0.3215)	0.2643 (0.2636)
90.4	−0.0121 (−0.0122)	0.3208 (0.3227)	0.2119 (0.2113)
90.3	−0.0120 (−0.0121)	0.3221 (0.3239)	0.1592 (0.1588)
90.2	−0.0119 (−0.0119)	0.3233 (0.3251)	0.1063 (0.1060)

The fixed model parameters: the exercise price $E = 100.0$, volatility $\sigma = 0.25$, dividend yield $q = 0.0$, maturity $\tau = 1.0$, interest rate $r = 0.1$, the lower barrier is set at 90.0, and the upper barrier is set at 140.0. The other parameters used by the function `dko_call` were: `nt` $= 200$, `ns_below_S0` $= 100$, `ns_above_S0` $= 100$, and $\Theta_m = 0.5$ (i.e., the Crank–Nicolson method). The results for $\Theta_m = 0.0$ (i.e., the implicit method) are shown in brackets; see Table 6, Boyle and Tian (1998).

Inspection of the results shows that the finite-difference grid method has both greater accuracy and faster convergence than the method proposed by BT. The

key to the accuracy achieved by `dko_call` is a combination of:

- The logarithmic transformation of the Black–Scholes equation
- The ability to place a grid line at both the upper barrier B_U, and also at the lower boundary B_L
- The use of a weighted Θ_m finite-difference scheme, $0 \leqslant \Theta_m \leqslant 1$, instead of the numerically unstable explicit finite-difference method used by a trinomial lattice which in *our notation* (see Section 5.4.2) is equivalent to $\Theta_m = 1$.

It should be mentioned that the function `dko_call` could, without much difficulty, be modified to deal with:

- American double knockout call options
- European double knockout put options
- American double knockout put options

and also a range of other variations which may include lockout periods, rebates, etc. In particular, options with time-varying barrier levels can be dealt with by using grid lines to locate the barrier position at each time instant.

5.5 Pricing American options using a stochastic lattice

In this section we consider the use of Monte Carlo simulation and stochastic lattices to price American options. Information on the use of Monte Carlo simulation to value both single asset and multiasset European options is provided in Chapter 4 and Chapter 6. The main difficulty in using simulation to value American options is the need to incorporate optimal early exercise policies. The standard simulation algorithms for valuing European contracts are *forward in time*. That is each price path, which contributes to the value of the option, is generated by stepping forward from current time, t, to option maturity, $t + \tau$, where τ is the duration of the option. For instance if there are n equispaced time steps of size Δt, and only one underlying asset, then we use the asset values $S_i, i = 0, \ldots, n$, where S_i corresponds to the asset value at the ith time instant, t_i, and $t_0 = t$. Here S_{i+1} is generated from the previous asset value S_i as follows:

$$\frac{S_{i+1}}{S_i} = \mathrm{d}S_i \quad \text{for } i = 0, \ldots, n-1 \tag{5.5.1}$$

where $\mathrm{d}S_i$ is a random variate taken from a *given* distribution. When S_i follows GBM, we have from Eq. (2.3.11) that:

$$\frac{S_{i+1}}{S_i} = \exp\left\{\left(r - \frac{\sigma_i^2}{2}\right)\Delta t + \sigma_i \,\mathrm{d}W_i\right\}, \quad i = 0, \ldots, n-1, \tag{5.5.2}$$

where $\mathrm{d}W_i \sim \mathrm{N}(0, \Delta t)$ and the usual definitions are used for σ_i and r.

For European exotic options (such as time dependent barrier options) the value of a particular price path will depend on the asset values $S_i, i = 0, \ldots, n$.

This is not true of European vanilla options whose value only depends on S_n, the underlying asset price at option maturity. The Monte Carlo approximation to the value of a European option is thus:

$$f = \frac{\sum_{j=1}^{n_{\text{sim}}} p_j(n_j)}{n_{\text{sim}}}$$

where n_{sim} is the number of simulations used, n_j is the number of time steps associated with the jth price path, and $p_j(n_j)$ is the value of the jth price path. In the case of European vanilla options we can use $n_j = 1$, $j = 1, \ldots, n_{\text{sim}}$; the accuracy obviously improves with increasing n_{sim}.

The valuation of American-style options, which include the possibility of early exercise, is more complicated. In Chapter 5 we described the use of binomial lattices to price American options when the underlying asset price process is GBM. Dynamic programming was used and the option prices were computed by working backward in time through the lattice. The application of Monte Carlo methods for pricing American options is described in Tilley (1993), Barraquand and Martineau (1995), and also Boyle, Broadie, and Glasserman (1997). Here we will outline the stochastic lattice approach discussed in Broadie and Glasserman (1997), where both a high estimator and a low estimator of the American option value are calculated. Since both of these biased estimators converge (with increasing number of simulations and lattice nodes) to the true option value, we will only consider how to compute the high estimator, θ_{H}. We summarize the approach as follows

- Set the parameters
- Generate the lattice asset prices
- Compute the lattice option prices
- Compute the Monte Carlo estimate.

We will now consider each of these steps in more detail.

Set the parameters

First we set the simulation parameters; that is: n_{sim} is the number of lattice simulations, b is the number of branches per lattice node, and d is the number of time instants in the lattice. Note: This definition of d here is different from that used in the original paper by Broadie and Glasserman (1997) where d is defined as the number of time steps in the lattice.

Generate the lattice asset prices

Next we generate the asset prices for the pth stochastic lattice. Since the lattice is non recombining at the i lattice time instant there are b^i nodes/asset prices. This contrasts with the binomial lattice of Chapter 5 where the asset prices at a given time step are arranged in ascending order, that is S_i^j increases with increasing j.

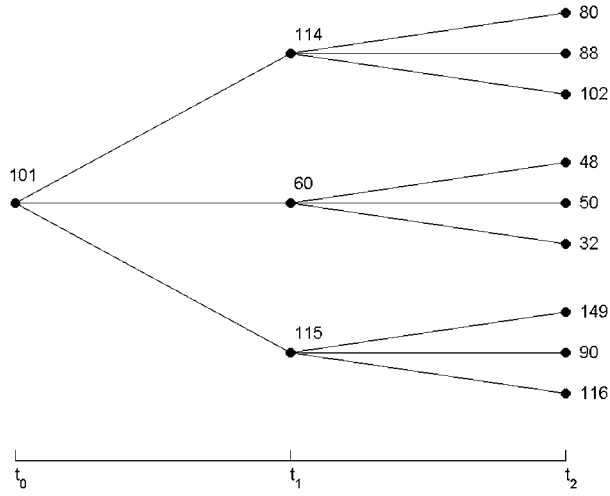

Figure 5.9 An example showing the asset prices generated for a stochastic lattice with three branches per node and two time steps, that is $b = 3$ and $d = 3$. The current asset value, 101, is at time t_0, and the asset values at option maturity are at time t_2.

We will denote the jth value at the ith time step by S_i^j. For example, in Fig. 5.9, where $b = 3$ and $d = 3$, we have for the first time step:

$$S_1^1 = 115, \qquad S_1^2 = 60, \quad \text{and } S_1^3 = 114$$

and for the second time step:

$$S_2^1 = 116, \qquad S_2^2 = 90, \qquad S_2^3 = 149, \qquad \ldots, \qquad S_2^7 = 102,$$
$$S_2^8 = 88, \qquad S_2^9 = 80$$

The kth asset price at the ith time step, S_i^k then generates the following asset prices at the $(i + 1)$th time step:

$$\frac{S_{i+1}^{(k-1)b+j}}{S_i^k} = \mathrm{d}S^j, \quad j = 1, \ldots, b, \ k = 1, \ldots, b^i,$$

where (see Eq. (5.5.1)), $\mathrm{d}S^j$ is a random variate from a *given* distribution. When S_i follows GBM, we therefore have:

$$\frac{S_{i+1}^{(k-1)b+j}}{S_i^k} = \exp\left\{\left(r - \frac{\sigma_i^2}{2}\right)\Delta t + \sigma_i \, \mathrm{d}W_i\right\}, \quad j = 1, \ldots, b, \ k = 1, \ldots, b^i$$

Compute the lattice option prices

The method used to compute the option values is *similar* to that used by the binomial lattice. The main difference is that there are now b branches per node instead of two. The option values are computed by starting at the lattice termi-

nal nodes and then iterating backward. Here we denote the kth option value at the ith time step by f_i^k.

The option values at the terminal nodes, time instant t_{d-1}, are computed in the usual manner. For a put we have:

$$f_{d-1}^k = \max\bigl(E - S_{d-1}^k, 0\bigr), \quad k = 1, \ldots, b^{d-1},$$

where E is the exercise price.

The option values at the $(i-1)$th time step are computed from those at the ith time step as follows:

$$f_{i-1}^k = \max\bigl(g_{i-1}^k, h_{i-1}^k\bigr)$$

where

$$h_{i-1}^k = \frac{\exp(-r\Delta t)}{b} \sum_{j=1}^{b} f_i^{(k-1)b+j}$$

and

$$g_{i-1}^k = \max\bigl(E - S_{i-1}^k, 0\bigr)$$

The option value for the pth stochastic lattice is therefore:

$$\theta_{\mathrm{H}}^p = f_0^1 = \frac{\exp(-r\Delta t)}{b} \sum_{j=1}^{b} f_1^j$$

Figure 5.10 shows the option values for an American call with strike price $E = 100$ and interest rate $r = 0$, when the lattice asset prices in Fig. 5.9 have been

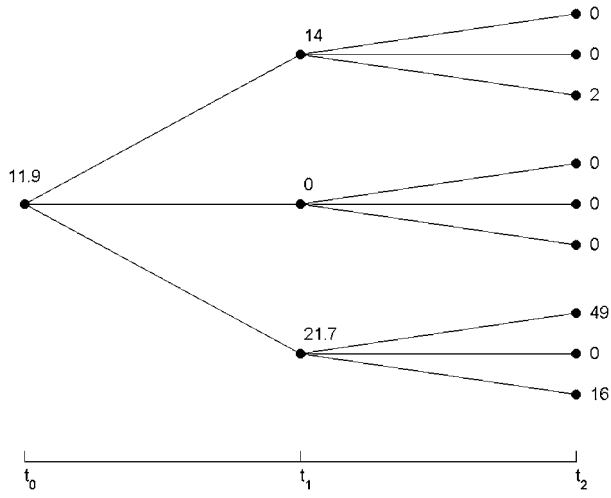

Figure 5.10 The option prices for the $b = 3, d = 3$ lattice in Fig. 5.9 corresponding to an American put with strike $E = 100$ and interest rate $r = 0$. The option values at the lattice nodes are computed backwards in time from the payoffs at maturity, t_2 to the current time t_0; the value of the option is 11.9.

used. To make things as clear as possible, we will show how the value of each node is computed.

Terminal nodes
The option values at the terminal nodes are:

$$f_2^1 = \max(116 - 100, 0) = 16, \qquad f_2^2 = \max(90 - 100, 0) = 0,$$
$$f_2^3 = \max(149 - 100, 0) = 49, \qquad f_2^4 = \max(32 - 100, 0) = 0,$$
$$f_2^5 = \max(50 - 100, 0) = 0, \qquad f_2^6 = \max(48 - 100, 0) = 0,$$
$$f_2^7 = \max(102 - 100, 0) = 2, \qquad f_2^8 = \max(88 - 100, 0) = 0,$$
$$f_2^9 = \max(80 - 100, 0) = 0$$

Time step 1
Here we have:

$$g_1^1 = \max(115 - 100, 0) = 15, \qquad g_1^2 = \max(60 - 100, 0) = 0,$$
$$g_1^3 = \max(114 - 100, 0) = 14$$

Since $r = 0$, we have $\exp(-r\Delta t) = 1$ which gives:

$$h_1^1 = \frac{1}{3}\left\{f_2^1 + f_2^2 + f_2^3\right\} = \frac{1}{3}\{16 + 0 + 49\} = 21.7$$

$$h_1^2 = \frac{1}{3}\left\{f_2^4 + f_2^5 + f_2^6\right\} = \frac{1}{3}\{0 + 0 + 0\} = 0$$

$$h_1^3 = \frac{1}{3}\left\{f_2^7 + f_2^8 + f_2^9\right\} = \frac{1}{3}\{2 + 0 + 0\} = 0.66$$

The option values are then computed as follows:

$$f_1^1 = \max\left(h_1^1, g_1^1\right) = \max(21.7, 15) = 21.7$$

$$f_1^2 = \max\left(h_1^2, g_1^2\right) = \max(0, 0) = 0$$

$$f_1^3 = \max\left(h_1^3, g_1^3\right) = \max(0.66, 14.0) = 14.0$$

Time step 0
Here

$$g_0^1 = \max(101 - 100, 0) = 1 \quad \text{and}$$
$$h_0^1 = \frac{1}{3}\left\{f_1^1 + f_1^2 + f_1^3\right\} = \frac{1}{3}\{21.7 + 0 + 14.0\} = 11.9$$

The final value of the option for this particular lattice is therefore:

$$f_1^1 = \max\left(h_0^1, g_0^1\right) = \max(11.9, 1) = 11.9$$

Compute the Monte Carlo estimate

The Monte Carlo estimate, θ_H, is computed as the average of $\theta_{\mathrm{H}}^{p}, p = 1, \ldots, n_{\mathrm{sim}}$, where n_{sim} is the number of simulations:

$$\theta_{\mathrm{H}} = \frac{\sum_{i=1}^{n_{\mathrm{sim}}} \theta_{\mathrm{H}}^{i}}{n_{\mathrm{sim}}}$$

In Code excerpt 5.21, we provide a computer program which prices single asset American put and call options using a stochastic lattice. The method used by the program is the *depth first* procedure outlined in Broadie and Glasserman (1997), which has the advantage that the memory requirements are only of order $b \times d$; as before, b is the number of branches per node and d is the number of time intervals.

Here it is assumed the underlying asset follows GBM and the function `normal(M, S)` is used to generate a normal distribution with mean `M` and standard deviation `S`. We can therefore check the accuracy of the simulation with that obtained by a closed form solution which assumes a lognormal asset distribution, in this case the formula in Geske and Johnson (1984).

However, the real power of this method is when the underlying asset follows a more realistic process which is non-Gaussian and time varying. The only modification to the code is to replace the call to `normal` with that of another probability distribution and supply the time-varying parameters to it.

```
// Stochastic lattice for computing the value of American and European options via Monte Carlo_
   simulation.
// Here we assume that the asset prices have a lognormal distribution, and so generate
// normal variates; this assumption can easily be removed.
 void __cdecl main()
{
  long i,j,jj,is_put,is_american,w[200],num_simulations,b,d,seed;
  double T,time_step,sqrt_time_step,opt_value,pay_off,log_fac,asset_price;
  double temp,opt_val,hold,sum_opt_val,disc;
  double tot_opt_vals, X, drift_term, std_term, S0, q, r, sigma, zero = 0.0;
  double v[200][60], opt_v[200][60];

  printf("Stochastic lattice for pricing European and American options \n");
  is_put = 1;      // If is_put == 0 then a call option, otherwise a put option
  T = 1.0;         // The time to maturity of the option
  is_american = 1; // If is_american  == 0 then an European option, otherwise an American_
                   option
  sigma = 0.2;     // The volatility of the underlying asset
  X = 110.0;       // The strike price
  S0 = 100.0;      // The current price of the underlying assset
  r = 0.1;         // The risk free interest rate
  q = 0.05;        // The continuous dividend yield
  d = 4;           // The number of time steps, the number time intervals = d - 1
  b = 50;          // The number of branches per node in the lattice
  time_step = T/(double)(d-1); // time step = T/(number of time intervals)
  sqrt_time_step = sqrt(time_step);
  disc = exp(-r*time_step); // The discount factor between time steps
  std_term = sigma*sqrt(time_step); // The standard deviation of each normal variate generated
  drift_term = (r - q - sigma*sigma*0.5)*time_step; // The mean value of each normal variate_
                                                    generated
  seed = 111;   // The seed for the random number generator
  srand(seed);
  tot_opt_vals = zero;
  num_simulations = 100;
  for (jj = 1; jj <= num_simulations; ++jj) {
    v[1][1] = S0;
```

Code excerpt 5.21.

```
w[1] = 1;
asset_price = S0;
for (j = 2; j <= d; ++j) {
  w[j] = 1;
  log_fac = normal(drift_term,std_term); // A normal variate:mean==drift_term,_
  standard deviation==std_term
  asset_price = asset_price*exp(log_fac); // Compute the new asset price: assuming a_
  lognormal distribution
  v[1][j] = asset_price;
}
j = d;
while (j > 0) {
  if ((j == d) && (w[j] < b)) { // CASE 1::Terminal node, set asset prices for b branches,_
  and option values for b-1 branches
    if (is_put ) {
       pay_off = MAX (X - v[w[j]][j],zero);
    }
    else {
       pay_off = MAX (v[w[j]][j]-X,zero);
    }
    opt_v[w[j]][j] = pay_off;
    asset_price = v[w[j-1]][j-1];
    log_fac = normal(drift_term,std_term);
    v[w[j]+1][j] = asset_price*exp(log_fac);
    w[j] = w[j] + 1;
  }
  else if ((j == d) && (w[j] == b)) { // CASE 2::Terminal node, set option value for last_
  branch
    if (is_put) {
      pay_off = MAX (X - v[w[j]][j],zero);
    }
    else {
      pay_off = MAX (v[w[j]][j]-X,zero);
    }
    opt_v[w[j]][j] = pay_off;
    w[j] = 0;
    j = j - 1;
  }
  else if ((j < d) && (w[j] < b)) { // CASE 3::Internal node, calculate option value for_
  node (parent wrt to cases 1 & 2)
    sum_opt_val = zero;           // Also generate a new terminal node and set asset_
                                     values.
    for (i = 1; i <= b; ++i) {
      sum_opt_val += opt_v[i][j+1];
    }
    temp = sum_opt_val/(double)b;
    hold = temp*disc;
    if (is_american) { // An American option
      if (is_put) {
        pay_off = MAX(X-v[w[j]][j],zero); // pay off for a put option
      }
      else {
         pay_off = MAX(v[w[j]][j]-X,zero);  // pay off for a call option
      }
      opt_val = MAX(pay_off,hold);
    }
    else { // A European option
      opt_val = hold;
    }
    opt_v[w[j]][j] = opt_val;
    if (j > 1) {
      asset_price = v[w[j-1]][j-1];
      log_fac = normal(drift_term,std_term);
      v[w[j]+1][j] = asset_price*exp(log_fac);
      w[j] = w[j] + 1;
      for (i = j + 1; i <= d; ++i) { // Generate a new terminal node
         log_fac = normal(drift_term,std_term);
         asset_price = asset_price*exp(log_fac);
         v[1][i] = asset_price;
         w[i] = 1;
      }
      j = d;
    }
    else {
      j = 0;
    }
```

Code excerpt 5.21 (*Continued*).

```
      }
      else if ((j < d) && (w[j] == b)) { // CASE 4::Internal node, calculate the option value_
                                            for the last branch
        sum_opt_val = zero;
        for (i = 1; i <= b; ++i) {
          sum_opt_val += opt_v[i][j+1];
        }
        temp = sum_opt_val/(double)b;
        hold = temp*disc;
        if (is_american) { // An American option
          if (is_put) {
            pay_off = MAX(X - v[w[j]][j],zero); // pay off for a put option
          }
          else {
            pay_off = MAX(v[w[j]][j]-X,zero); // pay off for a call option
          }
          opt_val = MAX(pay_off,hold);
        }
        else { // A European option
          opt_val = hold;
        }
        opt_v[w[j]][j] = opt_val;
        w[j] = 0;
        j = j - 1;
      }
    }
    tot_opt_vals = tot_opt_vals + opt_v[1][1]; // Sum the option values for each simulation
  }
  opt_value = tot_opt_vals/(double)num_simulations; // Compute the average option value
  printf ("The estimated option value = %12.4f\n", opt_value);
}
```

Code excerpt 5.21 A computer program that uses a stochastic lattice to value American and European options.

Table 5.13 American put option values, computed using the stochastic lattice given in Code excerpt 5.21, with four exercise times $t, t + \tau/3, t + 2\tau/3$ and $t + \tau$

X	MC_{50}^{100}	MC_{250}^{1}	True	Binomial lattice
70	0.118 (0.003)	0.123 (0.002)	0.121	0.126
80	0.663 (0.007)	0.672 (0.002)	0.670	0.696
90	2.317 (0.014)	2.307 (0.004)	2.303	2.389
100	5.830 (0.099)	5.720 (0.011)	5.731	5.928
110	11.564 (0.223)	11.361 (0.020)	11.341	11.770
120	20.205 (0.205)	20.000 (0.000)	20.000	20.052
130	30.054 (0.054)	30.000 (0.000)	30.000	30.000

The option parameters used were: $r = 0.1$, $q = 0.05$, $\tau = 1.0$, $\sigma = 0.2$, and $S = 100.0$. The column labelled MC_{50}^{100} refers to the results obtained using $d = 4$, $b = 50$, $num_simulations = 100$, and the column labelled MC_{250}^{1} refers to the results obtained using $d = 4$, $b = 50$, $num_simulations = 1$. The *true* values are those given in Broadie and Glasserman (1997), and were computed with the formula in Geske and Johnson (1984). The absolute error, $ABS(stochastic_lattice_value - true_value)$, is given in brackets. The last column gives the values computed using an accurate (6000 time step) binomial lattice.

In Table 5.13 we present computed values of an American put option with maturity τ, which can only be exercised at the following four times: $t, t + \tau/3$, $t + 2\tau/3$ and $t + \tau$, where t is the current time.

The column labelled MC_{50}^{100} contains the results obtained using 100 simulations of a stochastic lattice with 50 branches per node, and the column labelled MC_{250}^{1} contains the values computed using a single stochastic lattice with 250 branches per node. These values demonstrate that one high accuracy stochastic lattice can give better results than using the average of 100 lower accuracy lattices. In the last column we present the results obtained using a 6000 step binomial lattice in which it is possible to exercise the option at every time step. It can be seen that the binomial option values are higher than the *true* values, which only permit the option to be exercised at four distinct times. This is in agreement with the extra flexibility present in the binomial lattice.

6 Multiasset options

6.1 Introduction

In this section we consider the valuation of multiasset, *basket*, options within the Black–Scholes pricing framework. These options will be priced using the following techniques:

- Analytic methods
- Monte Carlo methods
- Multidimensional lattices

Analytic methods can be useful for pricing multiasset European options which have a known *closed form* solution. They are particularly appropriate for low dimensional European options, when the closed form expressions are not too difficult to evaluate.

Monte Carlo methods have the advantage that they can easily compute the value of multiasset European options, but have difficulty including the possibility of early exercise; this is required for American-style options.

On the other hand, multidimensional lattice techniques allow American options to be evaluated with ease. However, lattices become increasingly difficult to program as the number of dimensions increases, and the constraints of computer storage limits their use to problems involving (about) four or less assets.

6.2 The multiasset Black–Scholes equation

In Chapter 2 we mentioned that when the price, S, of a single asset follows geometric Brownian motion (GBM) the change in price, $\mathrm{d}S$, over a time interval, $\mathrm{d}t$, is given by:

$$\mathrm{d}S = rS\,\mathrm{d}t + \sigma S\,\mathrm{d}W$$

where r is the risk free interest rate, σ is the volatility of asset S, and $\mathrm{d}W \sim \mathrm{N}(0, \mathrm{d}t)$.

We also proved using Ito's lemma that the process followed by $Y = \log(S)$ is:

$$\mathrm{d}Y = \left(r - \frac{\sigma^2}{2}\right)\mathrm{d}t + \sigma\,\mathrm{d}W$$

where dY is the change in the value of $\log(S)$ over the time interval dt. Later on we derived the (Black–Scholes) partial differential equation that is satisfied by the value, V, of an option written on a single underlying asset. The equation is

$$\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

The above result can be generalized to deal with multiasset options. Suppose that m assets are described by the following processes:

$$dY_i = \left(r - \frac{\sigma_i^2}{2}\right)dt + \sigma_i\, dW_i, \quad i = 1, \ldots, m, \tag{6.2.1}$$

where the subscript i refers to the value associated with the ith asset. The m-element random vector dW is distributed according to $dW \sim N(0, C)$. The diagonal elements of C are $C_{ii} = \mathrm{Var}[dW_i] = dt$, $i = 1, \ldots, m$, and off-diagonal elements are:

$$C_{ij} = E[dW_i\, dW_j] = \rho_{i,j}\, dt, \quad i = 1, \ldots, m,\ j = 1, \ldots, m,\ i \neq j$$

We can also write the above equation in vector form by introducing the m-element vector dY which is normally distributed as:

$$dY \sim N(\nu, \overline{C}) \tag{6.2.2}$$

where ν is the mean vector and $\overline{C}$ is the covariance matrix. The elements of the covariance matrix are:

$$\begin{aligned} \overline{C}_{ii} &= \sigma_i^2\, dt, \quad i = 1, \ldots, m, \\ \overline{C}_{ij} &= \sigma_i \sigma_j \rho_{ij}\, dt, \quad i \neq j, i = 1, \ldots, m, j = 1, \ldots, m, \end{aligned} \tag{6.2.3}$$

where ρ_{ij} is the correlation coefficient between asset i and asset j; that is, the correlation between dW_i and dW_j. The elements of the mean vector ν are:

$$\nu_i = r - \frac{\sigma_i^2}{2}, \quad i = 1, \ldots, m \tag{6.2.4}$$

The value V of an option written on m assets satisfies the following partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 V}{\partial S_i\, \partial S_j} + r\sum_{i=1}^{m} S_i \frac{\partial V}{\partial S_i} - rV = 0.$$

For a European call on the maximum of m assets the pay-off $\mathcal{P}_c^{\mathrm{MAX}}$ at maturity (time τ) is given by $\mathcal{P}_c^{\mathrm{MAX}} = \max(\max(S_1^\tau, S_2^\tau, \ldots, S_m^\tau) - E, 0)$, where $S_i^\tau, i = 1, \ldots, m$, denotes the value of the ith asset at maturity, and E represents the strike price. Similarly a European put option on the minimum of m assets has a pay-off, $\mathcal{P}_p^{\mathrm{MIN}}$, at time τ, given by $\mathcal{P}_p^{\mathrm{MIN}} = \max(E - \min(S_1^\tau, S_2^\tau, \ldots, S_m^\tau), 0)$.

6.3 Multidimensional Monte Carlo methods

We have already mentioned that Monte Carlo simulation can easily price European multiasset options (also sometimes referred to as basket options, or rainbow options) involving a large number of assets (say 20 or more).

In addition Monte Carlo simulation can also include the following features into an option without much difficulty:

- Non-Gaussian distribution of stock returns; distributions with *heavy tails* are usually of interest because they more accurately represent what is observed in the financial markets
- Options with path dependency (such as barrier options, etc.); these are known as *exotic options*
- Complex time dependency (e.g., ARMA, GARCH or Levy processes) of model parameters such as interest rates, asset prices, etc.

The main drawbacks with Monte Carlo simulation are:

- It is to difficult compute the value of American-style options
- It is difficult (or impossible) to achieve the same accuracy that can be obtained using finite difference methods.

In a different section of this book we will show how Monte Carlo simulation can be used to price American options by using a hybrid *Monte Carlo lattice* approach originally developed by Boyle, Evnine, and Gibbs (1997).

In Chapter 3 we show that when pseudo-random numbers are used, the standard errors of integrals computed via Monte Carlo simulation decrease at the rate $N^{1/2}$, where N is the number of simulations. This means that it can require hundreds of thousands of simulations just to achieve an accuracy of 10^{-1} or 10^{-2} in the estimated option price. It is because of this that various Monte Carlo variance reduction techniques are used to increase the accuracy of the computed integral.

In this section we show how to price a three-asset basket option using Monte Carlo simulation; the accuracy of the results obtained with quasirandom numbers and pseudo-random numbers are compared.

The options we consider are European put and call options on the maximum and minimum of three assets. All the options have a maturity of one year, and the other model parameters used are given in Tables 6.1 and 6.2.

In Code excerpt 6.1 most of the work is done by the routine `multivariate_normal`. This generates a vector of multivariate pseudo-random numbers with a particular covariance matrix. In the program the values of the assets at current time t are $S_1 = S_2 = S_3 = 100$, and the option matures in one year.

The asset process evolves according to

$$\mathrm{d}Y_i = \log(S_{i,t+\mathrm{d}t}) - \log(S_{i,t}) = \left(r - \frac{\sigma_i^2}{2}\right)\mathrm{d}t + \sigma_i\,\mathrm{d}W_i, \quad i = 1, \ldots, m,$$

Table 6.1 The computed values and absolute errors, in brackets, for European options on the maximum of three assets

N_{sim}	Put	
	Quasi	Pseudo
500	0.890 (4.5948×10^{-2})	1.1044 (1.6839×10^{-1})
1000	0.924 (1.1534×10^{-2})	1.0193 (8.3297×10^{-2})
1500	0.919 (1.6807×10^{-2})	0.8957 (4.0344×10^{-2})
2000	0.932 (4.3221×10^{-3})	0.8995 (3.6488×10^{-2})
2500	0.932 (3.5698×10^{-3})	0.8886 (4.7352×10^{-2})
3000	0.937 (1.1376×10^{-3})	0.9025 (3.3548×10^{-2})
	Call	
	Quasi	Pseudo
500	22.629 (4.3231×10^{-2})	22.4089 (2.6312×10^{-1})
1000	22.683 (1.1306×10^{-2})	22.3520 (3.1998×10^{-1})
1500	22.670 (2.2954×10^{-3})	22.6346 (3.7430×10^{-2})
2000	22.685 (1.3299×10^{-2})	22.7675 (9.5491×10^{-2})
2500	22.670 (1.6619×10^{-3})	22.9326 (2.6058×10^{-1})
3000	22.679 (7.2766×10^{-3})	22.8050 (1.3301×10^{-1})

Monte Carlo simulation was used with both quasirandom (Sobol) sequences and pseudo-random sequences. The number of paths used varied from 500 to 3000. The parameters were: $E = 100.0, S_1 = S_2 = S_3 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2 = \sigma_3 = 0.2, \rho_{12} = \rho_{13} = \rho_{23} = 0.5, q_1 = q_2 = q_3 = 0.0$. The *accurate* values were 0.936 for a put and 22.672 for a call; see Table 6.7 and Table 2 of Boyle, Evnine, and Gibbs (1989).

where we have used the notation $S_{i,t}$ to denote the value of the ith asset at current time t, and $S_{i,t+\mathrm{d}t}$ to denote the value of the asset at the future time $t + \mathrm{d}t$. Simple rearrangement of the above equation gives:

$$\log\left(\frac{S_{i,t+\mathrm{d}t}}{S_{i,t}}\right) = \left(r - \frac{\sigma_i^2}{2}\right)\mathrm{d}t + \sigma_i\,\mathrm{d}W_i, \quad i = 1, \ldots, m$$

Taking exponentials of both sides we obtain:

$$\frac{S_{i,t+\mathrm{d}t}}{S_{i,t}} = \exp\left\{\left(r - \frac{\sigma_i^2}{2}\right)\mathrm{d}t + \sigma_i\,\mathrm{d}W_i\right\}, \quad i = 1, \ldots, m,$$

which is equivalent to:

$$S_{i,t+\mathrm{d}t} = S_{i,t}\exp\left\{\left(r - \frac{\sigma_i^2}{2}\right)\mathrm{d}t + \sigma_i\,\mathrm{d}W_i\right\} \tag{6.3.1}$$

Table 6.2 The computed values and absolute errors, in brackets, for European options on the minimum of three assets

N_{sim}	Put	
	Quasi	Pseudo
500	7.365 (3.8122×10^{-2})	7.6760 (2.7298×10^{-1})
1000	7.425 (2.1554×10^{-2})	7.7607 (3.5772×10^{-1})
1500	7.408 (5.1232×10^{-3})	7.5654 (1.6240×10^{-1})
2000	7.399 (3.6364×10^{-3})	7.4820 (7.8995×10^{-2})
2500	7.407 (4.1463×10^{-3})	7.3592 (4.3754×10^{-2})
3000	7.400 (2.7166×10^{-3})	7.3997 (3.3236×10^{-3})
	Call	
	Quasi	Pseudo
500	5.312 (6.3431×10^{-2})	5.3086 (5.9591×10^{-2})
1000	5.293 (4.3958×10^{-2})	5.4376 (1.8857×10^{-1})
1500	5.253 (4.0761×10^{-3})	5.4121 (1.6307×10^{-1})
2000	5.266 (1.7236×10^{-2})	5.4029 (1.5390×10^{-1})
2500	5.267 (1.7707×10^{-2})	5.4690 (2.2005×10^{-1})
3000	5.245 (3.5024×10^{-3})	5.4331 (1.8407×10^{-1})

Monte Carlo simulation was used with both quasirandom (Sobol) sequences and pseudo-random sequences. The number of paths used varied from 500 to 3000. The parameters were: $E = 100.0, S_1 = S_2 = S_3 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2 = \sigma_3 = 0.2, \rho_{12} = \rho_{13} = \rho_{23} = 0.5, q_1 = q_2 = q_3 = 0.0$. The *accurate* values were 7.403 for a put and 5.249 for a call; see Table 6.8 and Table 2 of Boyle, Evnine, and Gibbs (1989).

6.4 Introduction to multidimensional lattice methods

Finite-difference lattices can be used to value options on up to about 4 assets before they require impossibly large amounts of computer memory. The main advantage of finite-difference methods is that they are able to easily cater for American style early exercise within the option. This is not true of Monte Carlo methods. They can easily model complex European options, but have difficulty modelling American-style options.

In this section we use the approach of Kamrad and Ritchken (1991), and Boyle, Evnine, and Gibbs (1989), which we will call the BEGKR method, to price multiasset options. We first derive expressions for the jump size and jump probabilities for a single asset, and show that these are equivalent to those of the Cox, Ross, and Rubinstein binomial lattice (CRR lattice) discussed in Chapter 5. We will then give expressions for the jump sizes and jump probabilities of a general multiasset option.

To derive the BEGKR equations for one asset we first assume that the asset follows a lognormal process with drift $\mu = r - \sigma^2/2$, where r is the riskless interest rate and σ is the instantaneous volatility.

```
/* Monte Carlo simulation: 3 dimensional Black-Scholes, The results are compared with those_
   of Boyle et. al.,1989
   George Levy: 2007
*/
  long seed,i,num_simulations,iflag;
  double time_step,sqrt_time_step,rho, zero = 0.0, half = 0.5;
  double r,opt_val, opt_val1, tol;
  double the_max, the_min, X, ST1, ST2, ST3, ST4, S1, S2, S3, S4;
  double disc, sumit_max_put, sumit_max_call;
  double sumit_min_put, sumit_min_call;
  double *rvec = (double *)0;
  double rho_12, rho_13, rho_23;
  double  *c3, *c4, *z, *std, *means;
  double tmp1, tmp2, tmp3, tmp4, sigma1, sigma2, sigma3, sigma4;
  long is_fcall;

#define MEANS(I) means[(I)-1]
#define XBAR(I) xbar[(I)-1]
#define Z(I) z[(I)-1]
#define STD(I) std[(I)-1]
#define C3(I,J) c3[((I)-1) * 3 + ((J)-1)]

      seed = 111;
      r = 0.1;

      sigma1 = 0.2;
      sigma2 = 0.2;
      sigma3 = 0.2;
      S1 = 100.0;
      S2 = 100.0;
      S3 = 100.0;
      X = 100.0;

      rho_12 = 0.5;
      rho_13 = 0.5;
      rho_23 = 0.5;

      time_step = 1.0;
      sqrt_time_step = sqrt(time_step);
      disc = exp(-r*time_step);

      c3 = ALLOCATE(3*3, double);
      means = ALLOCATE(3, double);
      z = ALLOCATE(3, double);
      std = ALLOCATE(3, double);

      if ((!means) || (!std) || (!z) ) {
        printf("Allocation error \n");
       }

      tmp1 = sigma1*sigma1*time_step;
      tmp2 = sigma2*sigma2*time_step;
      tmp3 = sigma3*sigma3*time_step;

      C3(1,1) = tmp1;
      C3(2,2) = tmp2;
      C3(3,3) = tmp3;
      C3(1,2) = sigma1*sigma2*time_step*rho_12;
      C3(2,3) = sigma2*sigma3*time_step*rho_23;
      C3(1,3) = sigma1*sigma3*time_step*rho_13;
      C3(2,1) = C3(1,2);
      C3(3,1) = C3(1,3);
      C3(3,2) = C3(2,3);

      tmp1 = (r - sigma1*sigma1*half)*time_step;
      tmp2 = (r - sigma2*sigma2*half)*time_step;
      tmp3 = (r - sigma3*sigma3*half)*time_step;

      MEANS(1) = tmp1;
      MEANS(2) = tmp2;
      MEANS(3) = tmp3;
```

Code excerpt 6.1 A Monte Carlo simulation computer program, using pseudo-random numbers, for estimating the value of European put and call options on the maximum and minimum of three underlying assets. The results are presented in Tables 6.1 and 6.2.

```
      sumit_max_put = zero;
      sumit_max_call = zero;
      sumit_min_put = zero;
      sumit_min_call = zero;

      tol = 1.0e-8;

      srand(seed);
      is_fcall = 1;  /* initialisation call to the random number generator */
      multivariate_normal(is_fcall,&MEANS(1),3,&C3(1,1),3,tol,&rvec,&Z(1),&iflag);

      num_simulations = 6000;
      is_fcall = 0;
      for (i = 1; i <= num_simulations ; ++i) {

           /* continuation calls to the random number generator */
           multivariate_normal(is_fcall,&MEANS(1),3,&C3(1,1),3,tol,&rvec,&Z(1),&iflag);

           ST1 = S1*exp(Z(1));
           ST2 = S2*exp(Z(2));
           ST3 = S3*exp(Z(3));

// options on the maximum
           tmp2 = MAX(ST1,ST2);
           the_max = MAX(tmp2,ST3);
           tmp1 = the_max-X;
           opt_val1 = MAX(tmp1, zero);
           sumit_max_call += opt_val1*disc;

           tmp1 = X-the_max;
           opt_val1 = MAX(tmp1, zero);
           sumit_max_put += opt_val1*disc;

// options on the minimum
           tmp2 = MIN(ST1,ST2);
           the_min = MIN(tmp2,ST3);

           tmp1 = the_min-X;
           opt_val1 = MAX(tmp1, zero);
           sumit_min_call += opt_val1*disc;

           tmp1 = X-the_min;
           opt_val1 = MAX(tmp1, zero);
           sumit_min_put += opt_val1*disc;
     }

     opt_val = sumit_max_put/(double)num_simulations;  /* put option value on the maximum_
     of three assets */
     opt_val = sumit_max_call/(double)num_simulations; /* call option value on the maximum_
     of three assets */

     opt_val = sumit_min_put/(double)num_simulations; /* put option value on the minimum_
     of three assets */
     opt_val = sumit_min_call/(double)num_simulations; /* call option value on the maximum_
     of three assets */
}
```

Code excerpt 6.1 (*Continued*).

Therefore if S_t is the price of the asset at time t, and $S_{t+\Delta t}$ is the price at time instant $t + \Delta t$, we then have the following equations:

$$\log(S_{t+\Delta t}) = \log(S_t) + \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}\big(\mu\Delta t, \sigma^2\Delta t\big),$$

or equivalently

$$\log\left(\frac{S_{t+\Delta t}}{S_t}\right) \sim \mathrm{N}\big(\mu\Delta t, \sigma^2\Delta t\big)$$

where ε_t represents a random variable and as usual $\mathrm{N}(\mu\Delta t, \sigma^2\Delta t)$ denotes a Gaussian with mean $\mu\Delta t$ and variance $\sigma^2\Delta t$.

We will now consider the situation when ε_t either jumps up or down by an amount $\nu = \sigma\sqrt{\Delta t}$. For an up jump:

$$\log\left(\frac{S_{t+\Delta t}}{S_t}\right) = \sigma\sqrt{\Delta t}$$

and therefore $S_{t+\Delta t} = S_t \exp(\sigma\sqrt{\Delta t})$.

While for a down jump we have

$$\log\left(\frac{S_{t+\Delta t}}{S_t}\right) = -\sigma\sqrt{\Delta t}$$

and therefore $S_{t+\Delta t} = S_t \exp(-\sigma\sqrt{\Delta t})$.

The reader will notice that these expressions are the same as those for the CCR lattice of Chapter 5. That is: for an up jump $S_{t+\Delta t} = S_t u$, for a down jump $S_{t+\Delta t} = S_t d$, and $u = 1/d = \exp(\sigma\sqrt{\Delta t})$.

The probability of undergoing either an up or down jump occurring can be found by matching the mean and variance of ε_t.

From the mean:

$$E[\varepsilon_t] = \nu(p_u - p_d) = \mu\Delta t \tag{6.4.1}$$

and from the variance:

$$\text{Var}[\varepsilon_t] = \nu^2(p_u + p_d) = \sigma^2\Delta t \tag{6.4.2}$$

So combining Eqs. (6.4.1) and (6.4.2) we obtain:

$$\nu\mu\Delta t + \sigma^2\Delta t = 2\nu^2 p_u$$

so

$$p_u = \frac{1}{2}\left\{\frac{\sigma^2\Delta t}{\nu^2} + \frac{\mu\Delta t}{\nu}\right\}$$

Substituting $\nu = \sigma\sqrt{\Delta t}$ we obtain:

$$p_u = \frac{1}{2}\left\{1 + \frac{\mu\sqrt{\Delta t}}{\sigma}\right\} \tag{6.4.3}$$

and using the fact that $p_d = 1 - p_u$ gives:

$$p_d = \frac{1}{2}\left\{1 - \frac{\mu\sqrt{\Delta t}}{\sigma}\right\} \tag{6.4.4}$$

We shall now show that this is equivalent to the Cox–Rubinstein–Ross binomial model.

For the CRR model (Chapter 5, Eq. (5.3.19)) we have:

$$p_u = \frac{\exp(r\Delta t) - d}{u - d}$$

expanding $\exp(r\Delta t)$, u and d to order Δt we obtain:

$$\exp(r\Delta t) \sim 1 + r\Delta t$$

$$u = \exp\left(\sigma\sqrt{\Delta t}\right) \sim 1 + \sigma\sqrt{\Delta t} + \frac{\sigma^2}{2}\Delta t$$

$$d = \exp\left(\sigma\sqrt{\Delta t}\right) \sim 1 - \sigma\sqrt{\Delta t} + \frac{\sigma^2}{2}\Delta t$$

so

$$\exp(r\Delta t) - d \sim r\Delta t + \sigma\sqrt{\Delta t} - \frac{\sigma^2\Delta t}{2}$$

and

$$u - d \sim 2\sigma\sqrt{\Delta t}$$

So

$$p_u = \frac{\exp(r\Delta t) - d}{u - d} \sim \frac{r\Delta t + \sigma - \sigma^2/2\Delta t}{2\sigma\sqrt{\Delta t}}$$

which simplifies to

$$p_u = \frac{1}{2}\left\{1 + \frac{\mu\sqrt{\Delta t}}{\sigma}\right\}$$

and therefore

$$p_d = 1 - p_u = \frac{1}{2}\left\{1 - \frac{\mu\sqrt{\Delta t}}{\sigma}\right\}$$

which are the expressions for p_u and p_d given in Eqs. (6.4.1) and (6.4.2), respectively. So we have shown that, to first order in Δt, both the size of the jump and the probability of the jump are the same as the CRR binomial model.

The attractive feature of the BEGKR binomial lattice model is that it can easily be generalized to describe a model consisting of k assets. Here we will merely quote the results in Kamrad and Ritchken (1991). As before it is assumed that the asset prices follow a multivariate lognormal distribution. Let $\mu_i = r - \sigma_i^2/2$, and σ_i be the instantaneous mean and variance, respectively ($i = 1, 2, \ldots, k$) and let ρ_{ij} be the correlation between asset i and j. The binomial model now requires 2^k possible jumps in the time interval Δt. Let m denote the state of the process after time Δt with p_m representing the probability of state m ($m = 1, \ldots, 2^k$). The probabilities of these jumps are now given by:

$$p_m = \left\{1 + \sqrt{\Delta t}\sum_{i=1}^{k} x_{im}\left(\frac{\mu_i}{\sigma_i}\right) + \sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\left(x_{ij}^m\rho_{ij}\right)\right\},$$

$$m = 1, 2, \ldots, 2^k, k \geqslant 2,$$

where $x_{im} = 1$ if asset i has an up jump in state m, and $x_{im} = -1$ if asset i has a down jump in state m. In addition $x_{ij}^m = 1$ if asset i and asset j have jumps in the same direction in state m, and $x_{ij}^m = -1$ if asset i and asset j have jumps in the opposite direction in state m.

6.5 Two asset options

In this section we consider options based on the underlying prices of two assets, S_1 and S_2. We give analytic formulae to price European exchange options and also those based on the maximum or minimum of two assets. In addition we show how to construct binomial lattices for the valuation of two asset American-style options.

6.5.1 European exchange options

A European exchange option gives the holder the right to exchange one asset for another asset at maturity; see Margrabe (1978). Let the real-world processes of assets S^A and S^B be:

$$\begin{aligned} \mathrm{d}S_t^A &= S_t^A \mu_A \,\mathrm{d}t + S_t^A \sigma_A \,\mathrm{d}W_A^P \\ \mathrm{d}S_t^B &= S_t^B \mu_B \,\mathrm{d}t + S_t^B \sigma_B \,\mathrm{d}W_B^P \end{aligned}$$

where S_t^A denotes the value of asset A at time t and S_t^B denotes the value of asset B at time t—the other symbols have their *obvious* meanings.

We will now find the value, at current time t_0, of an option that gives the holder the right to exchange asset A for asset B at maturity T. The payoff at maturity is $H_T = \max(S_T^B - S_T^A, 0)$.

If we use the value of asset A as the numeraire then, from Eq. (4.2.1), the value of the exchange option at time t_0 is:

$$V(t_0) = S_{t_0}^A E^{\mathbb{Q}}\left[\frac{\max(S_T^B - S_T^B, 0)}{S_T^A}\right]$$

which can be written as

$$V(t_0) = S_{t_0}^A E^{\mathbb{Q}}\left[\max\left(\left(\frac{S_T^B}{S_T^A}\right) - 1, 0\right)\right] \tag{6.5.1}$$

where $\mathbb{Q}$ is the probability measure under which the relative price (S_t^B/S_t^A) is a martingale.

The process followed by (S_t^B/S_t^A) can be found by substituting $X_1 = S_t^B$ and $X_2 = S_t^A$ into Eq. (2.6.7). This yields

$$\begin{aligned} \mathrm{d}\left(\frac{S_t^B}{S_t^A}\right) = &\left(\frac{S_t^B}{S_t^A}\right)\left\{\mu_B - \mu_A + \sigma_A^2 - \sigma_A \sigma_B \rho_{AB}\right\} \mathrm{d}t \\ &+ \left(\frac{S_t^B}{S_t^A}\right)\left\{\sigma_B \,\mathrm{d}W_B^P - \sigma_A \,\mathrm{d}W_A^P\right\} \end{aligned}$$

Let $\widehat{X} = \sigma_B \,\mathrm{d}W_B^P - \sigma_A \,\mathrm{d}W_A^P$, so $E[\widehat{X}] = \sigma_B E[\mathrm{d}W_B^P] - \sigma_A E[\mathrm{d}W_A^P] = 0$, and $\mathrm{Var}[\widehat{X}] = \sigma_B^2 \,\mathrm{d}t + \sigma_A^2 \,\mathrm{d}t - 2\sigma_B \sigma_A \rho_{AB} \,\mathrm{d}t$ where we have used (see Appendix C.3):

$$\begin{aligned} &\mathrm{Var}[a \,\mathrm{d}W_1 + b \,\mathrm{d}W_2] = a^2 \,\mathrm{Var}[\mathrm{d}W_1] + b^2 \,\mathrm{Var}[\mathrm{d}W_2] + 2ab \,\mathrm{Cov}[\mathrm{d}W_1, \mathrm{d}W_2] \\ &\mathrm{Var}\left[\mathrm{d}W_B^P\right] = \mathrm{Var}\left[\mathrm{d}W_A^P\right] = \mathrm{d}t \quad \text{and} \quad \mathrm{Cov}\left[\mathrm{d}W_B^P, \mathrm{d}W_A^P\right] = \sigma_B \sigma_A \rho_{AB} \,\mathrm{d}t \end{aligned}$$

which means that $\widehat{X} \sim \mathrm{N}(0, \sigma_B^2\,\mathrm{d}t + \sigma_A^2\,\mathrm{d}t - 2\sigma_B\sigma_A\rho_{AB}\,\mathrm{d}t)$ and the variate $(\sigma_B^2 + \sigma_A^2 - 2\sigma_B\sigma_A\rho_{AB})\,\mathrm{d}W^P$ is from the same distribution as $\widehat{X}$.

Therefore we can write:

$$\mathrm{d}\left(\frac{S_t^B}{S_t^A}\right) = \left(\frac{S_t^B}{S_t^A}\right)\bar{\mu}\,\mathrm{d}t + \left(\frac{S_t^B}{S_t^A}\right)\bar{\sigma}\,\mathrm{d}W^P \tag{6.5.2}$$

where

$$\bar{\sigma} = \sqrt{\sigma_B^2 + \sigma_A^2 - 2\sigma_B\sigma_A}$$

and

$$\bar{\mu} = \mu_B - \mu_A + \sigma_A^2 - \sigma_B\sigma_A\rho_{AB}$$

Following Section 4.4.3 we choose the probability measure $\mathbb{Q}$ so that the drift term in Eq. (6.5.2) is zero. We have

$$\mathrm{d}W^P = \mathrm{d}W^Q - \left(\frac{\bar{\mu}}{\bar{\sigma}}\right)\mathrm{d}t$$

Substituting this into Eq. (6.5.2) gives

$$\mathrm{d}\left(\frac{S_t^B}{S_t^A}\right) = \left(\frac{S_t^B}{S_t^A}\right)\bar{\sigma}\,\mathrm{d}W^Q \tag{6.5.3}$$

It can be seen that Eq. (6.5.3) is identical to Eq. (4.4.31) but with the mapping:

$$S_t \to \left(\frac{S_t^B}{S_t^A}\right), \quad \sigma \to \bar{\sigma},\ r \to 0 \tag{6.5.4}$$

Now combining Eqs. (4.4.35) and (4.4.37) we have

$$\exp(-r\tau)E^{\mathbb{Q}}\big[\max(S_T - E, 0)\big] = SN_1(d_1) - E\exp(-r\tau)N_1(d_2)$$

where d_1 and d_2 have been defined in Section 4.4.3.

Therefore,

$$E^{\mathbb{Q}}\big[\max(S_T - 1, 0)\big] = \exp(r\tau)SN_1(d_1) - N_1(d_2)$$

Using the mapping defined in Eq. (6.5.4), and also $E = 1$ in the Black–Scholes formula, we have:

$$E^{\mathbb{Q}}\left[\left(\left(\frac{S_T^B}{S_T^A}\right) - 1, 0\right)\right] = \left(\frac{S_{t_0}^B}{S_{t_0}^A}\right)N_1(d_1) - N_1(d_2)$$

and from Eq. (6.5.1):

$$V(t_0) = S_{t_0}^A\left\{\left(\frac{S_{t_0}^B}{S_{t_0}^A}\right)N_1(d_1) - N_1(d_2)\right\}$$

This means that the value of the exchange option at time t_0 is:

$$V(t_0) = S_{t_0}^B N_1(d_1) - S_{t_0}^A N_1(d_2)$$

where

$$d_1 = \frac{\log(S^A_{t_0}/S^B_{t_0}) + \frac{1}{2}(T - t_0)\bar{\sigma}^2}{\bar{\sigma}\sqrt{T - t_0}}$$

$$d_2 = \frac{\log(S^A_{t_0}/S^B_{t_0}) - \frac{1}{2}(T - t_0)\bar{\sigma}^2}{\bar{\sigma}\sqrt{T - t_0}}$$

6.5.2 European options on the maximum or minimum

Here we present the results from Stulz (1982) and Johnson (1987) concerning the value of European put and call options on the maximum and minimum of two assets, see Code excerpts 6.2 and 6.3, and results in Tables 6.3 and 6.4.

```
void rainbow_bs_2d(double *opt_value, double S1, double S2, double X, double sigma1,
                   double sigma2, double rho, double opt_mat, double r, long is_max,_
                   long *iflag)
{
/* Input parameters:
   =================
   S1              - the current price of the underlying asset 1,
   S2              - the current price of the underlying asset 2,
   X               - the strike price,
   sigma1          - the volatility of asset 1,
   sigma2          - the volatility of asset 2,
   rho             - the correlation coefficient between asset 1 and asset 2,
   opt_mat         - the time to maturity,
   r               - the interest rate,
   is_max          - if is_max is 1 then the option is a call on the maximum of two assets,_
                     otherwise the option is a
                     call on the minimum of two assets.
   Output parameters:
   ==================
   opt_value       - the value of the option,
   iflag           - an error indicator.
*/

 double one=1.0,two=2.0,zero=0.0;
 double eps,d1,d2_1,d2_2,temp,temp1,temp2,pi,np;
 double rho_112, rho_212, d1_prime;
 double sigma, term1, term2, term3;
 long ifailx = 0;

      if(X < EPS) {   /*  ERROR the strike price is too small */
         *iflag = 2;
          return;
      }
      if (sigma1 < EPS) {   /*  ERROR the volatility (sigma1) is too small */
         *iflag = 3;
         return;
      }
      if (sigma2 < EPS) {   /* ERROR the volatility (sigma2) is too small */
         *flag = 3;
         return;
      }
      if (opt_mat < EPS) {  /* ERROR the time to maturity (opt_mat) is too small */
          *iflag = 3;
          return;
      }
      sigma = sqrt((sigma1*sigma1 + sigma2*sigma2) - two*sigma1*sigma2*rho);
```

Code excerpt 6.2 Function to calculate the value of a European call on the maximum or minimum of two assets using the analytic result of Johnson (1987) and Stulz (1982).

```
    if (is_max == 1) { /* then the maximum of two assets */
        /* calculate term1 */
        temp = log(S1/X);
        d1 = temp+(r+(sigma1*sigma1/two))*opt_mat;
        d1 = d1/(sigma1*sqrt(opt_mat));
        temp = log(S1/S2);
        d1_prime = temp+(sigma*sigma/two)*opt_mat;
        d1_prime = d1_prime/(sigma*sqrt(opt_mat));
        rho_112 = (sigma1 - rho*sigma2) / sigma;
        term1 =  cum_norm2(d1,d1_prime,rho_112,&ifailx);
        term1 =  term1*S1;
        /* calculate term2 */
        temp = log(S2/X);
        d1 = temp+(r+(sigma2*sigma2/two))*opt_mat;
        d1 = d1/(sigma2*sqrt(opt_mat));
        temp = log(S2/S1);
        d1_prime = temp+(sigma*sigma/two)*opt_mat;
        d1_prime = d1_prime/(sigma*sqrt(opt_mat));
        rho_212 = (sigma2 - rho*sigma1) / sigma;
        term2 =  S2*cum_norm2(d1,d1_prime,rho_212,&ifailx);
        /* calculate term3 */
        temp = log(S1/X);
        d2_1 = temp+(r-(sigma1*sigma1/two))*opt_mat;
        d2_1 = d2_1/(sigma1*sqrt(opt_mat));
        temp = log(S2/X);
        d2_2 = temp+(r-(sigma2*sigma2/two))*opt_mat;
        d2_2 = d2_2/(sigma2*sqrt(opt_mat));
        term3 =  one-cum_norm2(-d2_1,-d2_2,rho,&ifailx);
        *opt_value = term1+term2-X*exp(-r*opt_mat)*term3;
    }
    else {  /* the minimum of two assets */
        /* calculate term1 */
        temp = log(S1/X);
        d1 = temp+(r+(sigma1*sigma1/two))*opt_mat;
        d1 = d1/(sigma1*sqrt(opt_mat));
        temp = log(S1/S2);
        d1_prime = temp+(sigma*sigma/two)*opt_mat;
        d1_prime = d1_prime/(sigma*sqrt(opt_mat));
        rho_112 = (sigma1 - rho*sigma2) / sigma;
        term1 =  cum_norm2(d1,-d1_prime,-rho_112,&ifailx);
        term1 =  term1*S1;
        /* calculate term2 */
        temp = log(S2/X);
        d1 = temp+(r+(sigma2*sigma2/two))*opt_mat;
        d1 = d1/(sigma2*sqrt(opt_mat));
        temp = log(S2/S1);
        d1_prime = temp+(sigma*sigma/two)*opt_mat;
        d1_prime = d1_prime/(sigma*sqrt(opt_mat));
        rho_212 = (sigma2 - rho*sigma1) / sigma;
        term2 =  S2*cum_norm2(d1,-d1_prime,-rho_212,&ifailx);
        /* calculate term3 */
        temp = log(S1/X);
        d2_1 = temp+(r-(sigma1*sigma1/two))*opt_mat;
        d2_1 = d2_1/(sigma1*sqrt(opt_mat));
        temp = log(S2/X);
        d2_2 = temp+(r-(sigma2*sigma2/two))*opt_mat;
        d2_2 = d2_2/(sigma2*sqrt(opt_mat));
        term3 =  cum_norm2(d2_1,d2_2,rho,&ifailx);
        *opt_value = term1+term2-X*exp(-r*opt_mat)*term3;
    }
    return;
}
```

Code excerpt 6.2 (*Continued*).

Call options on the maximum and minimum of two assets

Let the value of a European call option on the minimum of two assets, S_1 and S_2, with strike price E, maturity τ, and correlation coefficient ρ, be denoted by $c_{\min}$. The value of the corresponding call option on the maximum of these assets will be represented by $c_{\max}$.

```
void opt_rainbow_bs_2d(double *opt_value, double S1, double S2, double X, double sigma1,
          double sigma2, double rho, double opt_mat, double r, long is_max, long putcall,_
          long *flag)
{
/* Input parameters:
   ================
   S1             - the current price of the underlying asset 1,
   S2             - the current price of the underlying asset 2,
   X              - the strike price,
   sigma1         - the volatility of asset 1,
   sigma2         - the volatility of asset 2,
   rho            - the correlation coefficient between asset 1 and asset 2,
   opt_mat        - the time to maturity,
   r              - the interest rate,
   is_max         - if is_max is 1 then the option is on the maximum of two assets,_
                    otherwise the option is on
                    the minimum of two assets,
   putcall        - if putcall is 0 then the option is a call, otherwise the option is a put.
   Output parameters:
   ==================
   opt_value      - the value of the option,
   iflag          - an error indicator.
*/

   double temp1;
   double temp2;
   double fac;
   double a_zero = 1.0e-6; /* approximate zero number to prevent overflow in rainbow_bs_2d */
   if (putcall) { /* a put option */
      fac = X*exp(-r*opt_mat);
      rainbow_bs_2d(&temp1, S1, S2, a_zero, sigma1, sigma2, rho, opt_mat, r, is_max, flag);
      rainbow_bs_2d(&temp2, S1, S2, X, sigma1, sigma2, rho, opt_mat, r, is_max, flag);
      *opt_value = fac - temp1 + temp2;
   } else { /* a call option */
      rainbow_bs_2d(opt_value, S1, S2, X, sigma1, sigma2, rho, opt_mat, r, is_max, flag);
   }
}
```

Code excerpt 6.3 Function to calculate the value of a European put or call on the maximum or minimum of two assets using the analytic result of Johnson (1987) and Stulz (1982).

Table 6.3 The computed values and absolute errors for European put and call options on the maximum of two assets

Time	Call			Put		
	Analytic	Lattice	Error	Analytic	Lattice	Error
0.1	6.45320	6.45245	7.4972×10^{-4}	0.01524	0.01451	7.3344×10^{-4}
0.2	6.96192	6.95953	2.3845×10^{-3}	0.08252	0.08001	2.5106×10^{-3}
0.3	7.49587	7.49376	2.1084×10^{-3}	0.15787	0.15580	2.0675×10^{-3}
0.4	8.03710	8.04022	3.1260×10^{-3}	0.22362	0.22680	3.1768×10^{-3}
0.5	8.57808	8.57916	1.0757×10^{-3}	0.27762	0.27683	7.8867×10^{-4}
0.6	9.11529	9.10809	7.2006×10^{-3}	0.32115	0.31872	2.4328×10^{-3}
0.7	9.64700	9.64838	1.3826×10^{-3}	0.35598	0.35714	1.1548×10^{-3}
0.8	10.17238	10.17663	4.2571×10^{-3}	0.38372	0.38711	3.3891×10^{-3}

The results were obtained using a binomial lattice and the analytic formula (Johnson (1987) and Stulz (1982)). The time to maturity of the option is varied from 0.1 years to 0.8 years. The parameters are: $E = 44.0, S_1 = 40.0, S_2 = 50.0, r = 0.1, \sigma_1 = 0.2, \sigma_2 = 0.2, q_1 = q_2 = 0.0, \rho = 0.5, n_steps = 50$.

Table 6.4 The computed values and absolute errors for European put and call options on the minimum of two assets

Time	Call			Put		
	Analytic	Lattice	Error	Analytic	Lattice	Error
0.1	0.10810	0.10753	5.7048×10^{-4}	3.67044	3.66993	5.0955×10^{-4}
0.2	0.40862	0.40781	8.1047×10^{-4}	3.54551	3.54514	3.6961×10^{-4}
0.3	0.74162	0.73418	7.4339×10^{-3}	3.47882	3.47206	6.7642×10^{-3}
0.4	1.06989	1.07299	3.1076×10^{-3}	3.43283	3.43715	4.3214×10^{-3}
0.5	1.38675	1.38909	2.3414×10^{-3}	3.39540	3.40159	6.1826×10^{-3}
0.6	1.69203	1.69025	1.7757×10^{-3}	3.36145	3.35775	3.6964×10^{-3}
0.7	1.98691	1.96939	1.7520×10^{-2}	3.32859	3.31517	1.3417×10^{-2}
0.8	2.27276	2.26274	1.0018×10^{-2}	3.29566	3.29157	4.0885×10^{-3}

The results were obtained using a binomial lattice and the analytic formula (Johnson (1987) and Stulz (1982)). The time to maturity of the option is varied from 0.1 years to 0.8 years. The parameters are: $E = 44.0$, $S_1 = 40.0$, $S_2 = 50.0$, $r = 0.1$, $\sigma_1 = 0.2$, $\sigma_2 = 0.2$, $q_1 = q_2 = 0.0$, $\rho = 0.5$, $n_steps = 50$.

Then, following Stulz (1982) and Johnson (1987), we have:

$$\begin{aligned} c_{\max} = {} & S_1 N_2\big(d_1(S_1, E, \sigma_1^2), d_1'(S_1, S_2, \sigma_*^2), \rho_1\big) \\ & + S_2 N_2\big(d_1(S_2, E, \sigma_2^2), d_1'(S_2, S_1, \sigma_*^2), \rho_2\big) \\ & - E \exp(-r\tau)\big\{1 - N_2\big(-d_2(S_1, E, \sigma_1^2), -d_2(S_2, E, \sigma_2^2), \rho\big)\big\} \end{aligned} \tag{6.5.5}$$

and

$$\begin{aligned} c_{\min} = {} & S_1 N_2\big(d_1(S_1, E, \sigma_1^2), -d_1'(S_1, S_2, \sigma_*^2), -\rho_1\big) \\ & + S_2 N_2\big(d_1(S_2, E, \sigma_2^2), -d_1'(S_2, S_1, \sigma_*^2), -\rho_2\big) \\ & - E \exp(-r\tau) N_2\big(d_2(S_1, E, \sigma_1^2), d_2(S_2, E, \sigma_2^2), \rho\big) \end{aligned} \tag{6.5.6}$$

where $N_2(a, b, \rho)$ is the bivariate cumulative normal. It gives the cumulative probability, in a standardized bivariate normal distribution, that the variables x_1 and x_2 satisfy $x_1 \leqslant a$ and $x_2 \leqslant b$ when the correlation coefficient between x_1 and x_2 is ρ—the value is computed using the routine `cum_norm2`. The other symbols are defined as follows:

$$\sigma_*^2 = \sigma_1^2 - 2\rho\sigma_1\sigma_2 + \sigma_2^2$$

$$d_1(S_i, E, \sigma_i^2) = \frac{\log(S_i/E) + (r + \sigma_i^2/2)\tau}{\sigma_i\sqrt{\tau}}, \quad i = 1, 2,$$

$$d_2(S_i, E, \sigma_i^2) = \frac{\log(S_i/E) + (r - \sigma_i^2/2)\tau}{\sigma_i\sqrt{\tau}}, \quad i = 1, 2,$$

$$d_1'(S_i, S_j, \sigma_*^2) = \frac{\log(S_i/S_j) + (\sigma_*^2/2)\tau}{\sigma_*\sqrt{\tau}} \quad \text{for } i = 1,\ j = 2, \text{ or } i = 2,\ j = 1$$

and

$$\rho_1 = \frac{\sigma_1 - \rho\sigma_2}{\sigma_*}, \qquad \rho_2 = \frac{\sigma_2 - \rho\sigma_1}{\sigma_*}$$

It can also be shown that:

$$c_{\max}(S_1, S_2, E, \tau) + c_{\min}(S_1, S_2, E, \tau) = c(S_1, E, \tau) + c(S_2, E, \tau) \qquad (6.5.7)$$

where $c(S, E, \tau)$ is the value of a vanilla European call.

We will now derive an expression for the value of the corresponding European put options.

Put options on the minimum of two assets

It will now be shown that the price of a European put option on the minimum of two assets, $p_{\min}(S_1, S_2, E, \tau)$ is:

$$p_{\min}(S_1, S_2, E, \tau) = E\exp(-r\tau) - c_{\min}(S_1, S_2, 0, \tau) + c_{\min}(S_1, S_2, E, \tau) \qquad (6.5.8)$$

where the meaning of the symbols has been previously defined.

This result can be proved by considering the following two investments:

Portfolio A: Purchase one put option on the minimum of S_1 and S_2 with exercise price E.

Portfolio B: Purchase one discount bond which pays E at maturity. Write (that is, sell) one option on the minimum of S_1 and S_2 with an exercise price of zero. Purchase one option on the minimum of S_1 and S_2 with exercise price E.

We now consider the values of these portfolios at option maturity, time τ.

If $\min(S_1, S_2) \geqslant E$

Portfolio A: pays zero
Portfolio B: Pays $E - \min(S_1, S_2) + \min(S_1, S_2) - E = 0$

If $\min(S - 1, S_2) = S_1 < E$

Portfolio A: Pays $E - S_1$
Portfolio B: Pays $E - S_1 + 0 = E - S_1$

If $\min(S_1, S_2) = S_2 < E$

Portfolio A: Pays $E - S_2$
Portfolio B: Pays $E - S_2 + 0 = E - S_2$

We have therefore shown that, under all possible circumstances, Portfolio A has the same value as Portfolio B. This means that Eq. (6.5.8) is true.

Put options on the maximum of two assets

It will now be shown that the price of a European put option on the maximum of two assets, $p_{\max}(S_1, S_2, E, \tau)$ is:

$$p_{\max}(S_1, S_2, E, \tau) = E\exp(-r\tau) - c_{\max}(S_1, S_2, 0, \tau) + c_{\max}(S_1, S_2, E, \tau) \tag{6.5.9}$$

where, as before, the meaning of the symbols has been previously defined.

This result can be proved by considering the following two investments:

PORTFOLIO A: Purchase one put option on the maximum of S_1 and S_2 with exercise price E.

PORTFOLIO B: Purchase one discount bond which pays E at maturity. Write (that is, sell) one option on the maximum of S_1 and S_2 with an exercise price of zero. Purchase one option on the maximum of S_1 and S_2 with exercise price E.

As before we now consider the values of these portfolios at option maturity, time τ.

If $\max(S_1, S_2) \geqslant E$

Portfolio A: Pays zero
Portfolio B: Pays $E - \max(S_1, S_2) + \max(S_1, S_2) - E = 0$

If $\max(S_1, S_2) = S_1 < E$

Portfolio A: Pays $E - S_1$
Portfolio B: Pays $E - S_1 + 0 = E - S_1$

If $\max(S_1, S_2) = S_2 < E$

Portfolio A: Pays $E - S_2$
Portfolio B: Pays $E - S_2 + 0 = E - S_2$

It therefore follows that, under all possible circumstances, Portfolio A has the same value as Portfolio B, and this means that Eq. (6.5.9) is true.

6.5.3 American options

We assume that the prices of asset 1 and asset 2 follow a lognormal process with drift terms of $\mu_1 = r - \sigma_1^2/2$ and $\mu_2 = r - \sigma_2^2/2$, respectively. As before, r is the riskless interest rate and σ_1 and σ_2 are the instantaneous volatilities of asset 1 and asset 2.

If we let $S_{1,t}$ and $S_{2.t}$ denote the respective prices of asset 1 and asset 2 at time t, then we can write:

$$\log(S_{1,t+\Delta t}) = \log(S_{1,t}) + \varepsilon_{1,t} \tag{6.5.10}$$

and

$$\log(S_{2,t+\Delta t}) = \log(S_{2,t}) + \varepsilon_{2,t} \tag{6.5.11}$$

where $\varepsilon_{1,t}$ is a random normal variable with mean $\mu_1 \Delta t$ and variance $\sigma_1^2 \Delta t$, and $\varepsilon_{2,t}$ is a random normal variable with mean $\mu_2 \Delta t$ and variance $\sigma_2^2 \Delta t$.

In the binomial lattice model, over the time interval Δt, the variate $\log(S_{1,t})$ is only allowed to jump up or down by an amount $\nu_1 = \sigma_1 \sqrt{\Delta t}$, and similarly the variate $\log(S_{2,t})$ is only permitted to jump up and down by the amount $\nu_2 = \sigma_2 \sqrt{\Delta t}$. We will denote the probability of both $\log(S_{1,t})$ and $\log(S_{2,t})$ having an up jump over Δt by p_{uu}, and the probability of $\log(S_{1,t})$ having an up jump and $\log(S_{2,t})$ having a down jump by p_{ud}, etc.

The mean values in Eqs. (5.15) and (5.16) then give

$$E[\varepsilon_{1,t}] = \nu_1(p_{uu} + p_{ud} - p_{dd} - p_{du}) = \mu_1 \Delta t \tag{6.5.12}$$

$$E[\varepsilon_{2,t}] = \nu_2(p_{uu} + p_{ud} - p_{dd} - p_{du}) = \mu_2 \Delta t \tag{6.5.13}$$

and the variance/covariance terms yields

$$\text{Var}[\varepsilon_{1,t}] = \nu_1^2(p_{uu} + p_{ud} + p_{dd} + p_{du}) = \sigma_1^2 \Delta t \tag{6.5.14}$$

$$\text{Var}[\varepsilon_{2,t}] = \nu_2^2(p_{uu} + p_{ud} + p_{dd} + p_{du}) = \sigma_2^2 \Delta t \tag{6.5.15}$$

$$E[\varepsilon_{1,t}\varepsilon_{2,t}] = \nu_1\nu_2(p_{uu} - p_{ud} + p_{dd} - p_{du}) = \rho\sigma_1\sigma_2 \Delta t \tag{6.5.16}$$

where ρ is the correlation coefficient between $\varepsilon_{1,t}$ and $\varepsilon_{2,t}$.

We therefore obtain:

$$\begin{aligned}
p_{uu} + p_{ud} - p_{dd} + p_{du} &= \frac{\mu_1 \sqrt{\Delta t}}{\sigma_1} \\
p_{uu} - p_{ud} - p_{dd} + p_{du} &= \frac{\mu_2 \sqrt{\Delta t}}{\sigma_2} \\
p_{uu} + p_{ud} + p_{dd} + p_{du} &= 1 \\
p_{uu} - p_{ud} + p_{dd} - p_{du} &= \rho
\end{aligned}$$

These lead to the following jump probabilities:

$$\begin{aligned}
p_{uu} &= \frac{1}{4}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right) + \rho\right\} \\
p_{ud} &= \frac{1}{4}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right) - \rho\right\} \\
p_{dd} &= \frac{1}{4}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right) + \rho\right\} \\
p_{du} &= \frac{1}{4}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right) - \rho\right\}
\end{aligned}$$

In Code excerpt 6.4, we provide the computer code for a standard binomial lattice which prices options on the maximum and minimum of two assets.

The parameter M is the number of time steps used, and the lattice is constructed under the assumption that M is even.

```
void standard_2D_binomial(double *value, double S1, double S2, double X,
                          double sigma1, double sigma2, double rho, double T,
                          double r, double q1, double q2, long put,
                          long M, long opt_type, long is_american, long *iflag)
{
/* Input parameters:
   =================
   S1                  - the current price of the underlying asset 1
   S2                  - the current price of the underlying asset 2
   X                   - the strike price
   sigma1              - the volatility of asset 1
   sigma2              - the volatility of asset 2
   rho                 - the correlation coefficient between asset 1 and asset 2
   T                   - the time to maturity
   r                   - the interest rate
   q1                  - the continuous dividend yield for asset 1
   q2                  - the continuous dividend yield for asset 2
   put                 - if put is 0 then a call option, otherwise a put option
   M                   - the number of time steps, the zeroth time step is the root node_
                         of the lattice
   opt_type            - if opt_type is 0 then an option on the maximum of two asset
                         otherwise an option on the minimum of two assets
   is_american         - if is_american is 0 then a European option, otherwise_
                         an American option
   Output parameters:
   ==================
   value               - the value of the option,
   iflag               - an error indicator.
*/
  double discount,t1,dt,d1,d2,u1,u2;
  long i,j,m1,n,iflagx,jj,ind;
  double zero=0.0,hold;
  double temp,ds1,ds2,dv1,dv2,h,tmp;
  double *s1, *s2, *v;
  double p[4];
  long P1,P2,tdv;
  double sqrt_dt, t, mu1, mu2, jp1, jp2;
  double one = 1.0, half = 0.5, quarter = 0.25;
  long v1;
  if (!((M+1)/2) == (M/2)) printf ("ERROR THE NUMBER OF TIME STEPS IS NOT EVEN \n");
  tdv = M + 1;
#define V(I,J) v[(I) * tdv + (J)]
#define UU  0
#define UD  1
#define DD  2
#define DU  3
  dt = T/(double)M;
  sqrt_dt = sqrt(dt);
  jp1 = sigma1*sqrt_dt;
  jp2 = sigma2*sqrt_dt;
  mu1 = r - q1 - sigma1*sigma1*half;
  mu2 = r - q2 - sigma2*sigma2*half;
  u1 = exp(jp1); /* assign the jump sizes */
  u2 = exp(jp2);
  d1 = exp(-jp1);
  d2 = exp(-jp2);
  p[UU] = quarter*(one + sqrt_dt * ((mu1/sigma1) + (mu2/sigma2)) +  rho); /* set up the jump_
                                                                             probabilities */
  p[UD] = quarter*(one + sqrt_dt * ((mu1/sigma1) - (mu2/sigma2)) - rho);
  p[DD] = quarter*(one + sqrt_dt * (-(mu1/sigma1) - (mu2/sigma2)) + rho);
  p[DU] = quarter*(one + sqrt_dt * (-(mu1/sigma1) + (mu2/sigma2)) - rho);
  for (i = 0; i < 4; ++i) {
    if ((p[i] < zero) || (p[i] > 1.0)) printf ("ERROR p out of range\n");
  }
  discount = exp(-r*dt);
```

Code excerpt 6.4 Function to calculate the value of a European put or call on the maximum or minimum of two assets using a standard binomial lattice.

```
  for (i = 0; i < 4; ++i) {
    p[i] = p[i]*discount;
  }
/* Allocate the arrays v[(M+1)*(M+1)], s1[2*M+1] and s2[2*M+1] */
      .    .    .
    s1[M] = S1; /* assign the 2*M+1 asset values for s1 */
    for (i = 1; i <= M; ++i) {
       s1[M+i] = u1*s1[M+i-1];
       s1[M-i] = d1*s1[M-i+1];
    }
    s2[M] = S2; /* assign the 2*M+1 asset values for s2 */
    for (i = 1; i <= M; ++i) {
       s2[M+i] = u2*s2[M+i-1];
       s2[M-i] = d2*s2[M-i+1];
    }
    P1 = 0;
    for (i = 0; i <= M; ++i) { /* Calculate the option values at maturity  */
       P2 = 0;
       for (j = 0; j <= M; ++j) {
          if (opt_type == 0) { /* Maximum of two assets */
             if (put) {
                V(i,j)  = MAX(X - MAX(s1[P1],s2[P2]),zero);
             }
             else {
                V(i,j)  = MAX(MAX(s1[P1],s2[P2])-X,zero);
             }
          }
          else {
             if (put) { /* Minimum of two assets */
                 V(i,j)  = MAX(X - MIN(s1[P1],s2[P2]),zero);
             }
             else {
                 V(i,j)  = MAX(MIN(s1[P1],s2[P2])-X,zero);
             }
          }
           P2 = P2 + 2;
        }
        P1 = P1 + 2;
     }
    for (m1 = M-1; m1 >= 0; --m1) { /* work backwards through the lattice_
                                       to calculate option value */
       P1 = M-m1;
       for (i = 0; i <= m1; ++i) {
           P2 = M-m1;
           for (j = 0; j <= m1; ++j) {
              hold = p[UD]*V(i+1,j) + p[UU]*V(i+1,j+1) + p[DU]*V(i,j+1) + p[DD]*V(i,j);
              if (is_american) {       /* An American option */
                 if (opt_type == 0) {  /* Maximum of two assets */
                    if (put)
                        V(i,j) = MAX(hold,X-MAX(s1[P1],s2[P2]));
                    else
                        V(i,j) = MAX(hold,MAX(s1[P1],s2[P2])-X);
                 }
                 else {                /* Minimum of two assets */
                    if (put)
                        V(i,j) = MAX(hold,X-MIN(s1[P1],s2[P2]));
                    else
                        V(i,j) = MAX(hold,MIN(s1[P1],s2[P2])-X);
                 }
              }
              else {
                 V(i,j) = hold;
              }
              P2 = P2 + 2;
           }
           P1 = P1 + 2;
       }
    }
    *value = V(0,0);
}
```

Code excerpt 6.4 (*Continued*).

6.6 Three asset options

For three asset options (see Code excerpt 6.5 and results in Tables 6.5–6.8), we have the following jump probabilities:

$$p_{uuu} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3}\right) + \rho_{12} + \rho_{13} + \rho_{23}\right\}$$

$$p_{uud} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3}\right) + \rho_{12} - \rho_{13} - \rho_{23}\right\}$$

$$p_{udu} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3}\right) - \rho_{12} + \rho_{13} - \rho_{23}\right\}$$

$$p_{udd} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3}\right) - \rho_{12} - \rho_{13} + \rho_{23}\right\}$$

$$p_{duu} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3}\right) - \rho_{12} - \rho_{13} + \rho_{23}\right\}$$

$$p_{dud} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3}\right) - \rho_{12} + \rho_{13} - \rho_{23}\right\}$$

$$p_{ddu} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3}\right) + \rho_{12} - \rho_{13} - \rho_{23}\right\}$$

$$p_{ddd} = \frac{1}{8}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3}\right) + \rho_{12} + \rho_{13} + \rho_{23}\right\}$$

```
void standard_3D_binomial(double *value, double S1, double S2, double S3, double X,
     double sigma1, double sigma2, double sigma3, double rho_12, double rho_13, double rho_23,
     double T, double r, long put, long M, long opt_type, long is_american, long *iflag)
{
/* Input parameters:
   ================
   S1              - the current price of the underlying asset 1
   S2              - the current price of the underlying asset 2
   S3              - the current price of the underlying asset 3
   X               - the strike price
   sigma1          - the volatility of asset 1
   sigma2          - the volatility of asset 2
   sigma3          - the volatility of asset 3
   rho_12          - the correlation coefficient between asset 1 and asset 2
   rho_13          - the correlation coefficient between asset 1 and asset 3
   rho_23          - the correlation coefficient between asset 2 and asset 3
   T               - the time to maturity
   r               - the interest rate
   put             - if put is 0 then a call option, otherwise a put option
   M               - the number of time steps, the zeroth time step is the root node_
                     of the lattice
   opt_type        - if opt_type is 0 then an option on the maximum of two asset
                     otherwise an option on the minimum of two assets
   is_american     - if is_american is 0 then a European option, otherwise an American option.
   Output parameters:
   ==================
   value           - the value of the option,
   iflag           - an error indicator.
```

Code excerpt 6.5 Standard 3-dimensional binomial lattice.

```
*/
  double discount,t1,dt,d1,d2,d3,u1,u2,u3;
  long i,j,k,m1,n,iflagx,jj,ind;
  double zero=0.0,hold;
  double temp,ds1,ds2,dv1,dv2,h,tmp,tmp1,tmp2;
  double *s1, *s2, *s3, *v;
  double p[9];
  long P1,P2,P3,tdv, tdv2;
  double sqrt_dt, t, mu1, mu2, mu3, jp1, jp2, jp3;
  double one = 1.0, half = 0.5, eighth = 0.125;
  long v1;
  if (!((M+1)/2) == (M/2)) printf ("ERROR THE NUMBER OF TIME STEPS IS NOT EVEN \n");
  tdv = M + 1;
  tdv2 = tdv*tdv;
#define V(I,J, K) v[(I) * tdv2 + (J)*tdv + (K)]
#define UUU  0
#define UUD  1
#define UDU  2
#define UDD  3
#define DUU  4
#define DUD  5
#define DDU  6
#define DDD  7
  dt = T/(double)M;
  sqrt_dt = sqrt(dt);
  jp1 = sigma1*sqrt_dt;
  jp2 = sigma2*sqrt_dt;
  jp3 = sigma3*sqrt_dt;
  mu1 = r - sigma1*sigma1*half;
  mu2 = r - sigma2*sigma2*half;
  mu3 = r - sigma3*sigma3*half;
  u1 = exp(jp1); /* assign the jump sizes */
  u2 = exp(jp2);
  u3 = exp(jp3);
  d1 = exp(-jp1);
  d2 = exp(-jp2);
  d3 = exp(-jp3);
/* set up the jump probabilities  */
  p[UUU] = eighth*(one + sqrt_dt * ((mu1/sigma1) + (mu2/sigma2)_
  + (mu3/sigma3)) + rho_12 + rho_13 + rho_23);
  p[UUD] = eighth*(one + sqrt_dt * ((mu1/sigma1) + (mu2/sigma2)_
  - (mu3/sigma3)) + rho_12 - rho_13 - rho_23);
  p[UDU] = eighth*(one + sqrt_dt * ((mu1/sigma1) - (mu2/sigma2)_
  + (mu3/sigma3)) - rho_12 + rho_13 - rho_23);

  p[UDD] = eighth*(one + sqrt_dt * ((mu1/sigma1) - (mu2/sigma2)_
  - (mu3/sigma3)) - rho_12 - rho_13 + rho_23);
  p[DUU] = eighth*(one + sqrt_dt * (-(mu1/sigma1) + (mu2/sigma2)_
  + (mu3/sigma3)) - rho_12 - rho_13 + rho_23);
  p[DUD] = eighth*(one + sqrt_dt * (-(mu1/sigma1) + (mu2/sigma2)_
  - (mu3/sigma3)) - rho_12 + rho_13 - rho_23);
  p[DDU] = eighth*(one + sqrt_dt * (-(mu1/sigma1) - (mu2/sigma2)_
  + (mu3/sigma3)) + rho_12 - rho_13 - rho_23);
  p[DDD] = eighth*(one + sqrt_dt * (-(mu1/sigma1) - (mu2/sigma2)_
  - (mu3/sigma3)) + rho_12 + rho_13 + rho_23);
  for (i = 0; i < 8; ++i) {
     if ((p[i] < zero) || (p[i] > 1.0)) printf ("ERROR p[%ld] = %12.4f out_
      of range\n",i, p[i]);
  }
  discount = exp(-r*dt);
  for (i = 0; i < 8; ++i) {
     p[i] = p[i]*discount;
  }
/* Allocate the arrays v[(M+1)*(M+1)*(M+1)], s1[2*M+1], s2[2*M+1], and s3[2*M+1] */
          .    .    .
    s1[M] = S1;
    for (i = 1; i <= M; ++i) { /* assign the 2*M+1 asset values for s1 */
       s1[M+i] = u1*s1[M+i-1];
       s1[M-i] = d1*s1[M-i+1];
    }
    s2[M] = S2;
```

Code excerpt 6.5 (*Continued*).

```
for (i = 1; i <= M; ++i) { /* assign the 2*M+1 asset values for s2 */
   s2[M+i] = u2*s2[M+i-1];
   s2[M-i] = d2*s2[M-i+1];
}
s3[M] = S3;
for (i = 1; i <= M; ++i) { /* assign the 2*M+1 asset values for s2 */
   s3[M+i] = u3*s3[M+i-1];
   s3[M-i] = d3*s3[M-i+1];
}
/* Calculate the option values at maturity  */
P1 = 0;
for (i = 0; i <= M; ++i) {
       P2 = 0;
   for (j = 0; j <= M; ++j) {

       P3 = 0;
       for (k = 0; k <= M; ++k) {
          if (put) { /* put */
            if (opt_type == 0) { /* Maximum of 3 assets */
               tmp = MAX(s1[P1],s2[P2]);
               V(i,j,k)  = MAX(X - MAX(tmp,s3[P3]),zero);
            }
            else if (opt_type == 1) { /* Minimum of 3 assets */
               tmp = MIN(s1[P1],s2[P2]);
               V(i,j,k)  = MAX(X - MIN(tmp,s3[P3]),zero);
            }
          }
          else { /* call */
            ** Insert call option code using the supplied put option code as a template **
          }
          P3 = P3 + 2;
       }
       P2 = P2 + 2;
   }
   P1 = P1 + 2;
}
for (m1 = M-1; m1 >= 0; --m1) {  /* work backwards through the lattice to calculate_
                                    the option value */
   P1 = M-m1;
   for (i = 0; i <= m1; ++i) {
      P2 = M-m1;
      for (j = 0; j <= m1; ++j) {
         P3 = M-m1;
         for (k = 0; k <= m1; ++k) {
            hold = p[UUU]*V(i+1,j+1,k+1) + p[UUD]*V(i+1,j+1,k) + p[UDU]*V(i+1,j,k+1)_
            + p[UDD]*V(i+1,j,k) + p[DUU]*V(i,j+1,k+1) + p[DUD]*V(i,j+1,k)_
            + p[DDU]*V(i,j,k+1) + p[DDD]*V(i,j,k);
             if (is_american) {
               if (put) {
               if (opt_type == 0) { /* Maximum of 3 assets */
                    tmp = MAX(s1[P1],s2[P2]);
                  if (opt_type == 0) { /* Maximum of 3 assets */
                      tmp = MAX(s1[P1],s2[P2]);
                      tmp1  = MAX(tmp,s3[P3]);
                      tmp2 = MAX(X-tmp1,hold);
                      V(i,j,k) = MAX(tmp2,zero);
                  }
                  else if (opt_type == 1) { /* Minimum of 3 assets */
                       tmp = MIN(s1[P1],s2[P2]);
                       tmp1 = MIN(tmp,s3[P3]);
                       tmp2 = MAX(X-tmp1,hold);
                       V(i,j,k)  = MAX(tmp2,zero);
                  }
               }
               else { /* call option */
                   ** Insert call option code using the supplied put option_
                      code as a template **
               }
          }
```

Code excerpt 6.5 (*Continued*).

```
                else { /* European option */
                   V(i,j,k) = hold;
                }
                P3 = P3 + 2;
             }
             P2 = P2 + 2;
          }
          P1 = P1 + 2;
       }
    }
    *value = V(0,0,0);
}
```

Code excerpt 6.5 (*Continued*).

Table 6.5 The computed values and absolute errors for European options on the maximum of three assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
10	0.9112	2.485×10^{-2}	21.8601	8.119×10^{-1}
20	0.9192	1.678×10^{-2}	22.2807	3.913×10^{-1}
30	0.9232	1.276×10^{-2}	22.4137	2.583×10^{-1}
40	0.9254	1.056×10^{-2}	22.4792	1.928×10^{-1}
50	0.9268	9.180×10^{-3}	22.5182	1.538×10^{-1}
60	0.9278	8.236×10^{-3}	22.5441	1.279×10^{-1}

A binomial lattice was used and we show how the accuracy of the results depends on the number of time steps. The parameters are: $E = 100.0, S_1 = S_2 = S_3 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2 = \sigma_3 = 0.2, \rho_{12} = \rho_{13} = \rho_{23} = 0.5, q_1 = q_2 = q_3 = 0.0$. The *accurate* values are 0.936 for a put and 22.672 for a call; see Table 2, Boyle, Evnine, and Gibbs (1989).

Table 6.6 The computed values and absolute errors for European options on the minimum of three assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
10	7.0759	3.271×10^{-1}	5.2072	4.176×10^{-2}
20	7.2402	1.628×10^{-1}	5.2263	2.269×10^{-2}
30	7.2953	1.077×10^{-1}	5.2334	1.560×10^{-2}
40	7.3229	8.015×10^{-2}	5.2371	1.192×10^{-2}
50	7.3394	6.357×10^{-2}	5.2393	9.665×10^{-2}
60	7.3505	5.251×10^{-2}	5.2409	8.143×10^{-3}

A binomial lattice was used and we show how the accuracy of the results depends on the number of time steps. The parameters are: $E = 100.0, S_1 = S_2 = S_3 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2 = \sigma_3 = 0.2, \rho_{12} = \rho_{13} = \rho_{23} = 0.5, q_1 = q_2 = q_3 = 0.0$. The *accurate* values are 7.403 for a put and 5.249 for a call; see Table 2, Boyle, Evnine, and Gibbs (1989).

Table 6.7 The computed values and absolute errors for European options on the maximum of three assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
10	0.0122	4.041×10^{-2}	27.3180	5.091×10^{-1}
20	0.0295	2.314×10^{-2}	27.5743	2.528×10^{-1}
30	0.0366	1.600×10^{-2}	27.6589	1.682×10^{-1}
40	0.0404	1.221×10^{-2}	27.7010	1.261×10^{-1}
50	0.0427	9.868×10^{-3}	27.7263	1.008×10^{-1}
60	0.0443	8.280×10^{-3}	27.7431	8.396×10^{-2}

A binomial lattice was used and we show how the accuracy depends on the number of time steps. The parameters are: $E = 100.0$, $S_1 = S_2 = S_3 = 100.0$, $r = 0.1$, $\tau = 1.0$, $\sigma_1 = \sigma_2 = \sigma_3 = 0.2$, $\rho_{12} = -0.5$, $\rho_{13} = -0.5$, $\rho_{23} = 0.5$, $q_1 = q_2 = q_3 = 0.0$. The *accurate* values are 0.0526 for a put and 27.8271 for a call, and were computed using Monte Carlo simulation with 10^7 paths.

Table 6.8 The computed values and absolute errors for European options on the minimum of three assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
10	8.9646	3.130×10^{-1}	1.4047	1.800×10^{-1}
20	9.1231	1.545×10^{-1}	1.4963	8.836×10^{-2}
30	9.1749	1.027×10^{-1}	1.5261	5.857×10^{-2}
40	9.2007	7.694×10^{-2}	1.5409	4.381×10^{-2}
50	9.2161	6.151×10^{-2}	1.5497	3.499×10^{-2}
60	9.2264	5.123×10^{-2}	1.5556	2.913×10^{-2}

A binomial lattice was used and we show how the accuracy depends on the number of time steps. The parameters are: $E = 100.0$, $S_1 = S_2 = S_3 = 100.0$, $r = 0.1$, $\tau = 1.0$, $\sigma_1 = \sigma_2 = \sigma_3 = 0.2$, $\rho_{12} = -0.5$, $\rho_{13} = -0.5$, $\rho_{23} = 0.5$, $q_1 = q_2 = q_3 = 0.0$. The *accurate* values are 9.2776 for a put and 1.5847 for a call, and were computed using Monte Carlo simulation with 10^7 paths.

6.7 Four asset options

The results for four assets are presented in Tables 6.9 and 6.10. We have the following jump probabilities:

$$p_{uuuu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) + \rho_{12} + \rho_{13} + \rho_{14} + \rho_{23} + \rho_{24} + \rho_{34}\right\}$$

$$p_{uuud} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) + \rho_{12} + \rho_{13} - \rho_{14} + \rho_{23} - \rho_{24} + \rho_{34}\right\}$$

$$p_{uudu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) + \rho_{12} - \rho_{13} + \rho_{14} - \rho_{23} + \rho_{24} - \rho_{34}\right\}$$

$$p_{uudd} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) + \rho_{12} - \rho_{13} - \rho_{14} - \rho_{23} - \rho_{24} + \rho_{34}\right\}$$

$$p_{uduu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) - \rho_{12} + \rho_{13} + \rho_{14} - \rho_{23} - \rho_{24} + \rho_{34}\right\}$$

$$p_{udud} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) - \rho_{12} + \rho_{13} - \rho_{14} - \rho_{23} + \rho_{24} - \rho_{34}\right\}$$

$$p_{uddu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) - \rho_{12} - \rho_{13} + \rho_{14} + \rho_{23} - \rho_{24} - \rho_{34}\right\}$$

$$p_{uddd} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) - \rho_{12} - \rho_{13} - \rho_{14} + \rho_{23} + \rho_{24} + \rho_{34}\right\}$$

$$p_{duuu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) - \rho_{12} - \rho_{13} - \rho_{14} + \rho_{23} + \rho_{24} + \rho_{34}\right\}$$

$$p_{duud} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) - \rho_{12} - \rho_{13} + \rho_{14} + \rho_{23} - \rho_{24} - \rho_{34}\right\}$$

$$p_{dudu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) - \rho_{12} + \rho_{13} - \rho_{14} - \rho_{23} + \rho_{24} - \rho_{34}\right\}$$

$$p_{dudd} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) - \rho_{12} + \rho_{13} + \rho_{14} - \rho_{23} - \rho_{24} + \rho_{34}\right\}$$

$$p_{dduu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) + \rho_{12} - \rho_{13} - \rho_{14} - \rho_{23} - \rho_{24} + \rho_{34}\right\}$$

$$p_{ddud} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} + \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) + \rho_{12} - \rho_{13} + \rho_{14} - \rho_{23} + \rho_{24} - \rho_{34}\right\}$$

$$p_{dddu} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} + \frac{\mu_4}{\sigma_4}\right) + \rho_{12} + \rho_{13} - \rho_{14} + \rho_{23} - \rho_{24} - \rho_{34}\right\}$$

$$p_{dddd} = \frac{1}{16}\left\{1 + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2} - \frac{\mu_3}{\sigma_3} - \frac{\mu_4}{\sigma_4}\right) + \rho_{12} + \rho_{13} - \rho_{14} + \rho_{23} + \rho_{24} + \rho_{34}\right\}$$

Table 6.9 The computed values and absolute errors for European options on the maximum of four assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
4	0.6548	2.386×10^{-2}	22.1403	3.096
8	0.6268	4.129×10^{-3}	23.8640	1.372
12	0.6246	6.275×10^{-3}	24.3630	8.733×10^{-1}
16	0.6251	5.836×10^{-3}	24.5934	6.429×10^{-1}
20	0.6257	5.167×10^{-3}	24.7270	5.093×10^{-1}
24	0.6263	4.570×10^{-3}	24.8144	4.219×10^{-1}
28	0.6268	4.074×10^{-3}	24.8762	3.601×10^{-1}
32	0.6272	3.665×10^{-3}	24.9222	3.141×10^{-1}

A binomial lattice was used and we show how the accuracy depends on the number of time steps. The parameters are: $E = 100.0, S_1 = S_2 = S_3 = S_4 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.2, \rho_{12} = 0.5, \rho_{13} = 0.5, \rho_{23} = 0.5, q_1 = q2 = q_3 = q_4 = 0.0$. The *accurate* values are 0.6309 for a put and 25.2363 for a call, and were computed using Monte Carlo simulation with 10^7 paths.

Table 6.10 The computed values and absolute errors for European options on the minimum of four assets

N steps	Put		Call	
	Computed value	Error	Computed value	Error
4	7.8274	7.120×10^{-1}	3.5676	4.986×10^{-1}
8	8.1571	3.823×10^{-1}	3.8528	2.134×10^{-1}
12	8.2794	2.600×10^{-1}	3.9300	1.362×10^{-1}
16	8.3429	1.965×10^{-1}	3.9659	1.003×10^{-1}
20	8.3815	1.579×10^{-1}	3.9868	7.944×10^{-2}
24	8.4075	1.319×10^{-1}	4.0004	6.577×10^{-2}
28	8.4262	1.132×10^{-1}	4.0101	5.612×10^{-2}
32	8.4402	9.920×10^{-2}	4.0173	4.894×10^{-2}

A binomial lattice was used and we show how the accuracy depends on the number of time steps. The parameters are: $E = 100.0, S_1 = S_2 = S_3 = S_4 = 100.0, r = 0.1, \tau = 1.0, \sigma_1 = \sigma_2\sigma_3 = \sigma_4 = 0.2, \rho_{12} = 0.5, \rho_{13} = 0.5, \rho_{23} = 0.5, q_1 = q_2 = q_3 = q_4 = 0.0$. The *accurate* values are 8.5394 for a put and 4.0662 for a call, and were computed using Monte Carlo simulation with 10^7 paths.

7 Other financial derivatives

7.1 Introduction

In the preceding sections of the book we have only dealt with the valuation of equity derivatives. We are now going to consider some of the other types of trades such as

- Interest rate derivatives
- Foreign exchange derivatives
- Credit derivatives

A selection of these trades will be used by the C# portfolio pricing example in Chapter 8.

7.2 Interest rate derivatives

It is not possible to make *real* profit without risk. For example, if we (*without risk*) invest £1 in a bank account, then allowing for interest, the total number of pounds at future time T will be $1 + \Delta I(t, T)$, where $\Delta I(t, T)$ is the amount of interest accrued from t to time T. Since our investment grew by the riskless interest rate, the *real* value which allows for inflation must still be £1, so:

$$DF(t, T)\{1 + \Delta I(t, T)\} = 1$$

where $DF(t, T)$ is the discount factor from t to T.

Continuously compounded spot rate

When continuous compounding is used $1 + \Delta I(t, T) = \exp\{R(t, T)(T - t)\}$, where $R(t, T)$ is the *annual* continuously compounded spot rate between times t *years* and T *years*. We thus have:

$$DF(t, T)\exp\{R(t, T)(T - t)\} = 1 \tag{7.2.1}$$

so the discount factor is given by

$$DF(t, T) = \exp\{-R(t, T)(T - t)\} \tag{7.2.2}$$

and the continuously compounded rate is

$$R(t, T) = -\frac{\log(DF(t, T))}{T - t} \tag{7.2.3}$$

Simply compounded spot rate

When simple compounding is used $\Delta I(t,T) = L(t,T)(T-t)$, where $L(t,T)$ is the simply compounded spot rate between time t and T. Thus,

$$DF(t,T)\big\{1 + L(t,T)(T-t)\big\} = 1$$

and so the simply compounded rate is:

$$L(t,T) = \frac{1}{T-t}\left\{\frac{1}{DF(t,T)} - 1\right\} \tag{7.2.4}$$

7.2.1 Forward rate agreement

A Forward Rate Agreement (FRA) is a contract between two counterparties (referred to here as A and B), in which one counterparty (say A) agrees to pay interest at the (variable) spot rate, while the other agrees to pay at a fixed interest rate. Let the agreement start at time T_s and end at the maturity T_m, at which time the counterparties settle the amount that is owed. If P is the principal then, at time T_m, the contract has the following value to A:

$$V(T_m) = P\big\{(T_m - T_s)K - L(T_s,T_m)(T_m - T_s)\big\} \tag{7.2.5}$$

where K is the agreed fixed rate, and $L(T_s,T_m)$ is the simply compounded rate between times T_s and T_m. From Eq. (7.2.4) we have:

$$L(T_s,T_m) = \frac{1}{T_m - T_s}\left\{\frac{1}{DF(T_s,T_m)} - 1\right\}$$

so

$$V(T_m) = P\left\{(T_m - T_s)K - \left(\frac{1}{DF(T_s,T_m)} - 1\right)\right\} \tag{7.2.6}$$

The value of the FRA to A at time $t \leqslant T_m$ is therefore $FRA(t) = DF(t,T_m)V(T_m)$ which means that

$$FRA(t) = DF(t,T_m)P\left\{(T_m - T_s)K - \left(\frac{1}{DF(T_s,T_m)} - 1\right)\right\} \tag{7.2.7}$$

Using $DF(t,T_m) = DF(t,T_s)DF(T_s,T_m)$ we can rewrite Eq. (7.2.7) as

$$FRA(t) = DF(t,T_m)P\left\{(T_m - T_s)K - \left(\frac{DF(t,T_s)}{DF(t,T_m)} - 1\right)\right\} \tag{7.2.8}$$

The value of K that sets $FRA(t)$ to zero is termed the time t forward rate between times T_s and T_m, and is here denoted by $F(t,T_s,T_m)$. From Eq. (7.2.8),

$$F(t,T_s,T_m) = \frac{1}{T_m - T_s}\left\{\frac{DF(t,T_s)}{DF(t,T_m)} - 1\right\} \tag{7.2.9}$$

Combining Eqs. (7.2.8) and (7.2.10) we can express the value of the FRA as:

$$FRA(t) = DF(t,T_m)P\tau\big\{K - F(t,T_s,T_s+\tau)\big\} \tag{7.2.10}$$

where $\tau = T_m - T_s$ is known as the *tenor* of the FRA, and T_s is the *reset time* for forward rate $F(t,T_s,T_s+\tau)$.

7.2.2 Interest rate swap

Interest Rate Swaps (IRS) are very common financial instruments—it is not unusual for 80 percent of the trades in a portfolio to be IRS deals. Here we will provide a description of some of the characteristics of interest rate swaps; more detail can be found in Hull (2003).

A *vanilla* IRS consists of a strip of FRA trades, each FRA starting when the previous FRA finishes. The maturity date of the IRS thus corresponds to the maturity date of the last FRA.

Let the start times of the FRAs be $t_i, i = 0, \ldots, n-1$, and the maturity times of the FRAs be $t_i, i = 1, \ldots, n$; note that the FRA start times correspond to the forward rate reset times, and the maturity times correspond to the FRA payment times. We will now adopt the (common) convention of calling the trades *swaplets*, so an IRS is made up of a number of swaplets.

When the counterparty A pays the fixed rate and receives the floating rate the trade (from A perspective) is termed a *payer* IRS. Alternatively, if A receives the fixed rate and pays the floating rate, then the trade is termed a *receiver* IRS.

The value of an IRS at time t, where $t_{k-1} < t < t_k$, will now be considered.

We will assume that discount factors used to compute the forward rates and those used to discount the coupon payments are associated with the same yield curve. Using Eq. (7.2.10) we have:

$$IRS(t) = CDF(t, t_k) + \sum_{i=k+1}^{n} DF(t, t_i) P \tau_i \left\{ K - F(t, t_{i-1}, t_{i-1} + \tau_i) \right\} \tag{7.2.11}$$

where C is the *next coupon payment* after current time t (this occurs at time t_k), and τ_i is the tenor of the ith swaplet which starts at time t_{i-1} and ends at time t_i.

Note that the next coupon payment C for the swaplet starting at time t_{k-1} and maturing at time t_j is already known with certainty at time t because the forward rate $F(t, t_{k-1}, t_k)$ was reset in the past; $t_{k-1} < t$.

We will now rewrite Eq. (7.2.11) as:

$$IRS(t) = CDF(t, t_k) + FXD(t) - FLT(t) \tag{7.2.12}$$

where $FXD(t)$, the time t value of the *fixed leg*, is:

$$FXD(t) = PK \sum_{i=k+1}^{n} DF(t, t_i) \tau_i \tag{7.2.13}$$

and $FLT(t)$, the time t value of the *floating leg*, is:

$$FLT(t) = P \sum_{i=k+1}^{n} DF(t, t_i) \tau_i F(t, t_{i-1}, t_{i-1} + \tau_i) \tag{7.2.14}$$

The floating leg

We will now evaluate Eq. (7.2.14). The floating leg coupon payment at time t_i will be denoted by C_i and has value:

$$C_i = PF(t, t_{i-1}, t_{i-1} + \tau_i)\tau_i$$

where

$$F(t, t_{i-1}, t_{i-1} + \tau_i) = \left\{ \frac{DF(t, t_{i-1})}{DF(t, t_{i-1} + \tau_i)} - 1 \right\} \frac{1}{\tau_i}$$

From Eq. (7.2.14) we thus have:

$$\begin{aligned} FLT(t) &= \sum_{i=k+1}^{n} C_i DF(t, t_i) \\ &= P \sum_{i=k+1}^{n} DF(t, t_i)\tau_i \left\{ \frac{DF(t, t_{i-1})}{DF(t, t_i)} - 1 \right\} \frac{1}{\tau_i} \\ &= P \sum_{i=k+1}^{n} \big\{ DF(t, t_{i-1}) - DF(t, t_i) \big\} \\ &= P \big\{ DF(t, t_k) - DF(t, t_{k+1}) + DF(t, t_{k+1}) - \cdots \\ &\qquad - DF(t, t_i) + DF(t, t_i) - \cdots - DF(t, t_n) \big\} \\ &= P \big\{ DF(t, t_k) - DF(t, t_n) \big\} \end{aligned}$$

and so the value of the floating leg is

$$FLT(t) = P \big\{ DF(t, t_k) - DF(t, t_n) \big\} \tag{7.2.15}$$

The swap rate

The time t swap rate SR_t is the value of the fixed rate K that makes the $IRS(t)$ zero. Thus, from Eqs. (7.2.12)–(7.2.14):

$$P \big\{ DF(t, t_k) - DF(t, t_n) \big\} - CDF(t, t_k) = PSR_t \sum_{i=k+1}^{n} DF(t, t_i)\tau_i \tag{7.2.16}$$

so

$$SR_t = \frac{\{DF(t, t_k) - DF(t, t_n)\} - \frac{C}{P} DF(t, t_k)}{\sum_{i=k+1}^{n} DF(t, t_i)\tau_i} \tag{7.2.17}$$

Amortization

So far we have assumed that the principal is fixed and set to the value P. We will now deal with the situation where the principal varies with time according to the following *amortization* schedule:

$$AM_i = P_{i-1} - P_i, \quad i = 0, \ldots, n-1, \tag{7.2.18}$$

where P_i is the value of the principal at time t_i and $P_0 = P$.

The value of the floating leg is now computed as:

$$\begin{aligned}
FLT(t) &= \sum_{i=k+1}^{n} P_{i-1} DF(t, t_i) \tau_i \left\{ \frac{DF(t, t_{i-1})}{DF(t, t_i)} - 1 \right\} \frac{1}{\tau_i} \\
&= \sum_{i=k+1}^{n} P_{i-1} \big\{ DF(t, t_{i-1}) - DF(t, t_i) \big\} \\
&= P_k DF(t, t_k) - P_k DF(t, t_{k+1}) \\
&\quad + P_{k+1} DF(t, t_{k+1}) - P_{k+1} DF(t, t_{k+2}) + \cdots \\
&\quad + P_{n-2} DF(t, t_{n-1}) - P_{n-1} DF(t, t_{n-1}) - P_{n-1} DF(t, t_n) \\
&= P_k DF(t, t_k) - DF(t, t_{k+1}) \{ P_k - P_{k+1} \} - P_{k+1} - \cdots \\
&\quad - DF(t, t_{n-2}) \{ P_{n-2} - P_{n-1} \} - P_{n-1} DF(t, t_n) \\
&= P_k DF(t, t_k) - P_{n-1} DF(t, t_n) + \sum_{i=k+1}^{n-1} AM_i DF(t, t_i)
\end{aligned} \tag{7.2.19}$$

and so the value of the floating leg is:

$$FLT(t) = P_k DF(t, t_k) - P_{n-1} DF(t, t_n) + \sum_{i=k+1}^{n-1} AM_i DF(t, t_i) \tag{7.2.20}$$

When there is no amortization ($P_k = P_{n-1}$ and $AM_i = 0, i = 0, \ldots, n-1$) then Eq. (7.2.20) reduces to Eq. (7.2.15).

Basis swap

This is very similar to an interest rate swap, but now there are two floating legs, each with their associated principal amount.

For example, floating leg 1 could be associated with the one month LIBOR (London Inter Bank Offer Rate) and have a schedule of monthly payments, while floating leg 2 could use the three month LIBOR rates and have quarterly payments. In this case, the forward rates and discount factors for leg 1 would be computed using the 1 month LIBOR yield curve and the forward rates and discount factors for leg 2 will be computed using the three month LIBOR yield curve.

We will use the subscripts 1 and 2 to denote quantities associated with legs 1 and 2 respectively. The payment times associated with leg 1 are $t_1^i, i = 1, \ldots, n_1$, while those for leg 2 are $t_2^i, i = 1, \ldots, n_2$, and (for this example) $n_1 = 3n_2$.

If counterparty A makes the quarterly payments (that is, receives the payments made on leg 1), then the time t value of the basis swap is:

$$BS(t) = C_1 DF_1(t, t_{k_1}) + FLT_1(t) - C_2 DF_2(t, t_{k_2}) - FLT_2(t) \tag{7.2.21}$$

where we have used similar notation to that used in Eq. (7.2.12), with

$$FLT_1(t) = P_1\big(DF_1(t, t_{k_1}) - DF_1(t, t_n)\big) \tag{7.2.22}$$

and

$$FLT_2(t) = P_2\big(DF_2(t, t_{k_2}) - DF_2(t, t_n)\big) \tag{7.2.23}$$

In Eqs. (7.2.22) and (7.2.23) P_1 is the principal for leg 1 and P_2 is the principal for leg 2. The time of the next coupon payment for leg 1 is t_{k_1}, while that for leg 2 is t_{k_2}: in addition we have used the fact that $t_{n_1} = t_{n_2} = t_n$.

We will now consider the case in which the basis swap has been traded at time t, and shall also assume that $C_1 = C_2 = 0$ and $t = t_{k_1} = t_{k_2}$. In addition, we will specify that *principal exchange* occurs at the start (time t) and end (time t_n) of the swap.

The cash flows associated with principal exchange at the start of the swap leg are in the *opposite* direction to those for the remainder of the swap leg; see Hull (2003). We have:

$$FLT_1(t) = P_1 - P_1DF_1(t, t_n) + \big\{-P_1 + P_1DF(t, t_n)\big\} \tag{7.2.24}$$

and

$$FLT_2(t) = P_2 - P_2DF_2(t, t_n) + \big\{-P_2 + P_2DF(t, t_n)\big\} \tag{7.2.25}$$

where the principal exchange terms are in the curly brackets, and use discount factors $DF(t, T)$ derived from the *main currency* yield curve (in this case GBP) rather than $DF_1(t, T)$ or $DF_2(t, T)$.

It can be seen from Eq. (7.2.21) that principal exchange at the start of the swap causes leg 2 to contribute the *positive* amount P_2 to the value of the swap, while leg 1 contributes the *negative* amount P_2 to the value of the swap. In contrast, principal exchange at the end of the swap results in leg 2 contributing the *negative* amount $-P_2DF(t, t_n)$ to the swap value, while leg 1 contributes the *positive* amount $P_1DF(t, t_n)$. If $P_1 = P_2 = P$ then principal exchange does not affect the value of the basis swap. It can also be seen from Eq. (7.2.24) that if leg 1 used the *main* GBP yield curve instead of the one month LIBOR curve, then $DF_1(t, t_n) = DF(t, t_n)$ which would result in $FLT_1(t) = 0$. Similarly $DF_2(t, t_n) = DF(t, t_n)$ would mean that $FLT_2(t) = 0$.

If the valuation time t is after the trade has started, then Eq. (7.2.21) can be used to price the basis swap, but Eqs. (7.2.24) and (7.2.25) need to be modified as follows:

$$FLT_1(t) = P_1DF_1(t, t_{k_1}) - P_1DF_1(t, t_n) + \big\{P_1DF(t, t_n)\big\} \tag{7.2.26}$$

and

$$FLT_2(t) = P_2DF_2(t, t_{k_2}) - P_2DF_2(t, t_n) + \big\{P_2DF(t, t_n)\big\} \tag{7.2.27}$$

We will now consider how the timing of the coupon payment in relation to its associated forward rate affects the *present value*, V_{t_0}, of the coupon.

Coupon payment on time

In this section we will justify the approach we have adopted in obtaining the present value of future cashflows generated from *vanilla* forward rates.

From Eq. (4.2.1) we know that the value at time t_0 of a coupon payment at time t_k is:

$$V_{t_0} = DF(t_0, t_k) E^{\mathbb{Q}^k}\left[\frac{F(t_{k-1}, t_{k-1}, t_k)}{DF(t_k, t_k)}\right]\tau P$$

where the symbols have their usual meaning, and we have chosen the numeraire to be the zero coupon bond which matures at time t_k. Since $DF(t_k, t_k) = 1$ we can write

$$V_{t_0} = DF(t_0, t_k) E^{\mathbb{Q}^k}\big[F(t_{k-1}, t_{k-1}, t_k)\big]\tau P \tag{7.2.28}$$

In Section 7.2.3 we show that $F(t, t_{k-1}, t_k)$ follows the process:

$$\mathrm{d}\big(F(t, t_{k-1}, t_k)\big) = \sigma_k F(t, t_{k-1}, t_k)\,\mathrm{d}W^k \tag{7.2.29}$$

If we assume that σ_k is constant, then Eq. (7.2.29) is GBM (see Chapter 2) and has the solution:

$$F(t, t_{k-1}, t_k) = F(t_0, t_{k-1}, t_k)\exp\left(-\frac{(t - t_0)\sigma_k^2}{2} + \sigma_k W_t^k\right) \tag{7.2.30}$$

where we have taken $W_{t_0}^k = 0$.

Substituting $t = t_{k-1}$ into Eq. (7.2.30) gives:

$$F(t_{k-1}, t_{k-1}, t_k) = F(t_0, t_{k-1}, t_k)\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k W_{t_{k-1}}^k\right) \tag{7.2.31}$$

which means that:

$$\begin{aligned}
&E^{\mathbb{Q}^k}\big[F(t_{k-1}, t_{k-1}, t_k)\big] \\
&\quad = E^{\mathbb{Q}^k}\left[F(t_0, t_{k-1}, t_k)\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k W_{t_{k-1}}^k\right)\right] \\
&\quad = F(t_0, t_{k-1}, t_k) E^{\mathbb{Q}^k}\left[\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k W_{t_{k-1}}^k\right)\right] \\
&\quad = F(t_0, t_{k-1}, t_k) E^{\mathbb{Q}^k}\left[\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k\sqrt{t_{k-1} - t_0}\,\mathrm{N}(0, 1)\right)\right] \\
&\quad = F(t_0, t_{k-1}, t_k)
\end{aligned}$$

where we have used the fact (see Appendix D.2) that

$$\begin{aligned}
&E^{\mathbb{Q}^k}\left[\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k\sqrt{t_{k-1} - t_0}\,\mathrm{N}(0, 1)\right)\right] \\
&\quad = \exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \frac{(t_{k-1} - t_0)\sigma_k^2}{2}\right) = 1
\end{aligned}$$

Substituting for $E^{\mathbb{Q}^k}[F(t_{k-1}, t_{k-1}, t_k)]$ in Eq. (7.2.28):

$$V_{t_0} = P\tau F(t_0, t_{k-1}, t_k) = DF(t_0, t_k)\frac{1}{\tau}\left(\frac{DF(t_0, t_{k-1})}{DF(t_0, t_k)} - 1\right)P\tau$$

This yields

$$V_{t_0} = P\big\{DF(t_0, t_{k-1}) - DF(t_0, t_k)\big\} \tag{7.2.32}$$

which is our current method of valuing the future coupons generated by forward rates.

General payment timing

For the general case, in which the coupon payment date does not correspond to the end of its associated forward rate, we use the result from Eq. (7.2.43) that

$$\mathrm{d}W^k = \left(\frac{\mu_k}{\sigma_k}\right)\mathrm{d}t + \mathrm{d}W^i, \quad i \neq k \tag{7.2.33}$$

Equation (7.2.33) states that Brownian motion W^i under numeraire $DF(t, t_i)$ can be transformed into Brownian motion W^k under numeraire $DF(t, t_k)$ by the addition of a drift term—more detail can be found in Section 7.2.3. If we assume constant drift μ_k, $W^k_{t_0} = W^i_{t_0} = 0$ we obtain:

$$W^k_t = (t - t_0)\left(\frac{\mu_k}{\sigma_k}\right) + W^i_t \tag{7.2.34}$$

and in Eq. (7.2.30)

$$\sigma_k W^k_t = (t - t_0)\mu_k + \sigma_k W^i_t$$

A constant μ_k can be achieved by *freezing* the forward rates that make up μ_k; for example, $F(t, t_{k-1}, t_k) \rightarrow F(t_0, t_{k-1}, t_k)$; see Section 7.2.3 for more details concerning μ_k.

This means that $F(t, t_{k-1}, t_k)$ follows the process:

$$\mathrm{d}\big(F(t, t_{k-1}, t_k)\big) = F(t, t_{k-1}, t_k)\mu_k\,\mathrm{d}t + F(t, t_{k-1}, t_k)\sigma_k\,\mathrm{d}W^i$$

The above equation is GBM with drift and can be solved by modifying Eq. (7.2.30) to:

$$F(t, t_{k-1}, t_k) = F(t_0, t_{k-1}, t_k)\exp\left((t - t_0)\mu_k - \frac{(t - t_0)\sigma_k^2}{2} + \sigma_k W^i_t\right) \tag{7.2.35}$$

As before, the time t_0 value of the coupon payment at time t_i is:

$$V_{t_0} = DF(t_0, t_i)E^{\mathbb{Q}^i}\left[\frac{F(t_{k-1}, t_{k-1}, t_k)}{DF(t_i, t_i)}\right]\tau P$$

which since $DF(t_i, t_i) = 1$ becomes

$$V_{t_0} = DF(t_0, t_i)E^{\mathbb{Q}^i}\big[F(t_{k-1}, t_{k-1}, t_k)\big]\tau P \tag{7.2.36}$$

Now, from Eq. (7.2.35) we have:

$$\begin{aligned}
&E^{\mathbb{Q}^i}\big[F(t_{k-1}, t_{k-1}, t_k)\big] \\
&\quad = E^{\mathbb{Q}^i}\left[F(t_0, t_{k-1}, t_k)\exp\left((t - t_0)\mu_k - \frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k W^i_{t_{k-1}}\right)\right] \\
&\quad = F(t_0, t_{k-1}, t_k)\exp\big((t_{k-1} - t_0)\mu_k\big) \\
&\qquad \times E^{\mathbb{Q}^i}\left[\exp\left(-\frac{(t_{k-1} - t_0)\sigma_k^2}{2} + \sigma_k W^i_{t_{k-1}}\right)\right] \\
&\quad = F(t_0, t_{k-1}, t_k)\exp\big((t_{k-1} - t_0)\mu_k\big) \qquad (7.2.37)
\end{aligned}$$

where, as before, we have used the expectation given in Appendix D.2.

By expanding Eq. (7.2.37) to first order we obtain:

$$\begin{aligned}
&E^{\mathbb{Q}^i}\big[F(t_{k-1}, t_{k-1}, t_k)\big] \\
&\quad = F(t_0, t_{k-1}, t_k) + F(t_0, t_{k-1}, t_k)(t_{k-1} - t_0)\mu_k \qquad (7.2.38)
\end{aligned}$$

Substituting Eq. (7.2.38) into Eq. (7.2.36), we obtain a general expression for the value of the coupon payment:

$$V_{t_0} = DF(t_0, t_i)\tau\big\{F(t_0, t_{k-1}, t_k) + F(t_0, t_{k-1}, t_k)(t_{k-1} - t_0)\mu_k\big\} \qquad (7.2.39)$$

We will now consider the cases of early and late coupon payments.

Early coupon payment

Let us consider the case when $i = k - 1$. From Eq. (7.2.50):

$$\mu_k = \frac{\tau\sigma_k^2 F(t, t_{k-1}, t_k)}{1 + \tau F(t, t_{k-1}, t_k)}$$

First we freeze the forward rates in μ_k so we use the following:

$$\mu_k = \frac{\tau\sigma_k^2 F(t_0, t_{k-1}, t_k)}{1 + \tau F(t_0, t_{k-1}, t_k)}$$

Substituting for μ_k in Eq. (7.2.39) we obtain:

$$V_{t_0} = DF(t_0, t_{k-1})\tau P\left\{F(t_0, t_{k-1}, t_k) + (t_{k-1} - t_0)\frac{\sigma_k^2\tau_k F^2(t_0, t_{k-1}, t_k)}{1 + \tau F(t_0, t_{k-1}, t_k)}\right\}$$

(see Brigo and Mercurio, 2001, p. 387, and Hull, 2003).

Late coupon payment

We consider the case when $i = k + 1$. From Eq. (7.2.66):

$$\mu_k = -\frac{\tau\rho_{k,k+1}\sigma_k\sigma_{k+1}F(t, t_k, t_{k+1})}{1 + \tau F(t, t_k, t_{k+1})}$$

Freezing the forward rates we obtain:

$$\mu_k = -\frac{\tau\rho_{k,k+1}\sigma_k\sigma_{k+1}F(t_0,t_k,t_{k+1})}{1+\tau F(t_0,t_k,t_{k+1})}$$

Substituting for μ_k in Eq. (7.2.39) we obtain:

$$\begin{aligned} V_{t_0} = {} & DF(t_0,t_{k+1}) \\ & \times \tau P\Bigg\{F(t_0,t_{k-1},t_k) \\ & \quad -(t_k-t_0)\frac{\tau\rho_{k,k+1}\sigma_k\sigma_{k+1}F(t_0,t_{k-1},t_k)F(t_0,t_k,t_{k+1})}{1+\tau F(t_0,t_k,t_{k+1})}\Bigg\} \end{aligned}$$

7.2.3 Timing adjustment

In this section we derive expressions for the drift of the forward rate $F(t,t_{k-1},t_k)$ under various probability measures. We will denote the time t value of a zero coupon bond which pays 1 unit of currency at maturity t_i by $DF(t,t_i)$. For convenience we will also use the shortened notation $F_k = F(t,t_{k-1},t_k)$ and $DF_i = DF(t,t_i)$.

The probability measure under which all tradable assets are priced relative to the zero coupon bond price DF_i (that is, DF_i is the numeraire) will be denoted by $\mathbb{Q}^i$; under this probability measure the relative prices will be martingales. We will also denote Brownian motion under probability measure $\mathbb{Q}^i$ by W^i.

Case $i = k$

Here the maturity of the numeraire DF_k is at the expiry of the forward rate F_k.

Since $DF_k(1+\tau F_k)$ is a tradable its relative price, $(1+\tau F_k)/DF_k$, is a martingale under $\mathbb{Q}^k$ and thus has zero drift.

Also $DF_k(1+\tau F_k)/DF_k = 1+\tau F_k$ and, since both τ and 1 are constants, F_k must be a martingale under $\mathbb{Q}^k$. Thus the process for F_k has zero drift and is:

$$\mathrm{d}F_k = F_k\sigma_k\,\mathrm{d}W^k \tag{7.2.40}$$

For the general case in which $i \neq k$, the process followed by F_k is:

$$\mathrm{d}F_k = F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^i \tag{7.2.41}$$

where μ_k is a drift that needs to be determined. Equation (7.2.41) can be rewritten as:

$$\mathrm{d}F_k = F_k\sigma_k\left(\frac{\mu_k}{\sigma_k}\,\mathrm{d}t + \mathrm{d}W^i\right) \tag{7.2.42}$$

Comparing Eqs. (7.2.41) and (7.2.42) we have:

$$\mathrm{d}W^k = \frac{\mu_k}{\sigma_k}\,\mathrm{d}t + \mathrm{d}W^i \tag{7.2.43}$$

This equation gives the relation between Brownian motions under probability measures $\mathbb{Q}^k$ and $\mathbb{Q}^i$. We will now show how to compute the value of μ_k.

Case $i < k$

Here we consider situations in which the maturity of the numeraire DF_i is before the expiry of the forward rate F_k.

$i = k - 1$

In this case DF_{k-1} is the numeraire, the forward rate is $F_k = F(t, t_{k-1}, t_k)$, and the numeraire matures at time t_{k-1}, while the forward rate has expiry t_k.

Since DF_k is a tradable the relative price, $\phi = DF_k/DF_{k-1}$, is a martingale under $\mathbb{Q}^{k-1}$, and thus has zero drift.

Now

$$\phi = \frac{DF_k}{DF_{k-1}} = \frac{1}{1+\tau F_k} \tag{7.2.44}$$

where we have used:

$$DF_{k-1} = DF_k(1+\tau F_k) \tag{7.2.45}$$

Let the stochastic process followed by F_k under $\mathbb{Q}^{k-1}$ be:

$$\mathrm{d}F_k = F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k-1} \tag{7.2.46}$$

and the drift, μ_k, is to be determined.

Using Ito we have:

$$\mathrm{d}\phi = \frac{\partial\phi}{\partial F_k}\,\mathrm{d}F_k + \frac{1}{2}\frac{\partial^2\phi}{\partial F_k^2}E\big[(\mathrm{d}F_k)^2\big], \quad E\big[(\mathrm{d}F_k)^2\big] = \sigma_k^2F_k^2\,\mathrm{d}t, \tag{7.2.47}$$

where from Eq. (7.2.44):

$$\frac{\partial\phi}{\partial F_k} = -\frac{\tau}{(1+\tau F_k)^2}, \qquad \frac{\partial^2\phi}{\partial {F_k}^2} = \frac{2\tau^2}{(1+\tau F_k)^3} \tag{7.2.48}$$

Substituting the values in Eq. (7.2.48) into Eq. (7.2.47) we obtain:

$$\mathrm{d}\phi = -\frac{\tau\phi}{1+\tau F_k}\big\{F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k-1}\big\} + \frac{1}{2}\frac{\tau^2 2\phi\sigma^2F_k^2\,\mathrm{d}t}{(1+\tau F_k)^2}$$

which can be rearranged as:

$$\mathrm{d}\phi = \left\{-\frac{\tau F_k\mu_k\phi}{1+\tau F_k} + \frac{\phi\tau^2\sigma_k^2F_k^2}{(1+\tau F_k)^2}\right\}\mathrm{d}t - \frac{\tau\phi F_k\sigma_k}{1+\tau F_k}\,\mathrm{d}W^{k-1} \tag{7.2.49}$$

Now since Eq. (7.2.49) is driftless:

$$\left\{-\frac{\tau F_k\mu_k\phi}{1+\tau F_k} + \frac{\phi\tau^2\sigma_k^2F_k^2}{(1+\tau F_k)^2}\right\}\mathrm{d}t = 0$$

and

$$\mu_k = \frac{\tau\sigma_k^2F_k}{1+\tau F_k} \tag{7.2.50}$$

Substituting Eq. (7.2.50) into Eq. (7.2.46):

$$\mathrm{d}F_k = \frac{\tau\sigma_k^2 F_k^2}{1+F_k\tau}\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k-1}$$

or

$$\mathrm{d}F_k = F_k\sigma_k\left\{\frac{\tau\sigma_k^2 F_k}{1+F_k\tau}\,\mathrm{d}t + \mathrm{d}W^{k-1}\right\} \tag{7.2.51}$$

Comparing Eqs. (7.2.51) and (7.2.40) thus yields:

$$\mathrm{d}W^k = \frac{\tau\sigma_k F_k}{1+F_k\tau}\,\mathrm{d}t + \mathrm{d}W^{k-1} \tag{7.2.52}$$

which is the relationship between the Brownian motions $\mathrm{d}W^{k-1}$ and $\mathrm{d}W^k$ under the respective probability measures $\mathbb{Q}^{k-1}$ and $\mathbb{Q}^k$.

$i \leqslant k-2$

Let the stochastic process followed by F_k under $\mathbb{Q}^{k-2}$ be:

$$\mathrm{d}F_k = F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k-2} \tag{7.2.53}$$

where W^{k-2} is Brownian motion under probability measure $\mathbb{Q}^{k-2}$, and the drift, μ_k, is unknown.

Replacing k with $k-1$ in Eq. (7.2.52) gives:

$$\mathrm{d}W^{k-1} = \frac{\tau\sigma_{k-1}F_{k-1}}{1+F_{k-1}\tau}\,\mathrm{d}t + \mathrm{d}W^{k-2} \tag{7.2.54}$$

and using Eq. (7.2.54) to substitute for $\mathrm{d}W^{k-1}$ in Eq. (7.2.52) we obtain:

$$\mathrm{d}W^k = \frac{\tau\sigma_k F_k}{1+F_k\tau}\,\mathrm{d}t + \frac{\tau\sigma_{k-1}F_{k-1}}{1+F_{k-1}\tau}\,\mathrm{d}t + \mathrm{d}W^{k-2} \tag{7.2.55}$$

Replacing $\mathrm{d}W^k$ in Eq. (7.2.40) with that given in Eq. (7.2.55):

$$\mathrm{d}F_k = F_k\sigma_k\left\{\frac{\tau\sigma_k F_k}{1+F_k\tau}\,\mathrm{d}t + \frac{\tau\sigma_{k-1}F_{k-1}}{1+F_{k-1}\tau}\,\mathrm{d}t\right\} + F_k\sigma_k\,\mathrm{d}W^{k-2} \tag{7.2.56}$$

so the drift is:

$$\mu_k = \frac{\tau\sigma_k^2 F_k}{1+\tau F_k} + \frac{\tau F_{k-1}\sigma_k\sigma_{k-1}\rho_{k,k-1}}{1+\tau F_{k-1}} \tag{7.2.57}$$

The following general expression can be derived in a similar manner:

$$\mathrm{d}F_k = \sigma_k F_k \sum_{j=i+1}^{k} \frac{\rho_{k,j}\tau\sigma_j F_j}{1+\tau F_j}\,\mathrm{d}t + \sigma_k F_k\,\mathrm{d}W^i \tag{7.2.58}$$

where all the symbols have the same meanings as before, but now i can take any integer value less than k.

Case $i > k$

We now consider the case when the maturity of the numeraire DF_i is after the expiry of the forward rate F_k.

$i = k + 1$

Here DF_{k+1} is the numeraire and F_k is the forward rate which starts at time t_{k-1} and ends at time t_k.

Since DF_{k-1} is a tradable its relative price, $\phi = DF_{k-1}/DF_{k+1}$, is a martingale under $\mathbb{Q}^{k+1}$, and thus has zero drift.

Now:

$$\phi = \frac{DF_{k-1}}{DF_{k+1}} = (1 + \tau F_k)(1 + \tau F_{k+1}) \tag{7.2.59}$$

where the processes for F_k and F_{k+1} are

$$\mathrm{d}F_{k+1} = F_{k+1}\sigma_{k+1}\mathrm{d}W^{k+1} \tag{7.2.60}$$

$$\mathrm{d}F_k = F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k+1} \tag{7.2.61}$$

and the drift, μ_k, is to be determined.

Using Ito we have:

$$\mathrm{d}\phi = \frac{\partial\phi}{\partial F_k}\mathrm{d}F_k + \frac{\partial\phi}{\partial F_{k+1}}\mathrm{d}F_{k+1} + \frac{1}{2}\sum_{i=k}^{k+1}\sum_{j=k}^{k+1}\frac{\partial^2\phi}{\partial F_i\,\partial F_j}E[\mathrm{d}F_i, \mathrm{d}F_j] \tag{7.2.62}$$

where

$$\begin{aligned}
&\frac{\partial\phi}{\partial F_k} = \tau(1 + \tau F_{k+1}), \qquad \frac{\partial\phi}{\partial F_{k+1}} = \tau(1 + \tau F_k)\\
&\frac{\partial^2\phi}{\partial F_k^2} = \frac{\partial^2\phi}{\partial F_{k+1}^2} = 0, \qquad \frac{\partial^2\phi}{\partial F_{k+1}\,\partial F_k} = \frac{\partial^2\phi}{\partial F_k\,\partial F_{k+1}} = \tau^2\\
&E[\mathrm{d}F_k, \mathrm{d}F_{k+1}] = [\mathrm{d}F_{k+1}, \mathrm{d}F_k] = \rho_{k,k+1}\sigma_k\sigma_{k+1}F_kF_{k+1}\,\mathrm{d}t
\end{aligned} \tag{7.2.63}$$

Substituting the values in (7.2.63) into Eq. (7.2.62) we obtain:

$$\mathrm{d}\phi = \tau(1 + \tau F_{k+1})\,\mathrm{d}F_k + \tau(1 + \tau F_k)\,\mathrm{d}F_{k+1} + \tau^2\rho_{k,k+1}\sigma_k\sigma_{k+1}F_kF_{k+1}\,\mathrm{d}t$$

After expanding the terms in $\mathrm{d}F_k$ and $\mathrm{d}F_{k+1}$ we have:

$$\begin{aligned}
\mathrm{d}\phi = {} & \tau(1 + \tau F_{k+1})\left\{F_k\mu_k\,\mathrm{d}t + F_k\sigma_k\,\mathrm{d}W^{k+1}\right\}\\
& + \tau(1 + \tau F_k)F_{k+1}\sigma_{k+1}\,\mathrm{d}W^{k+1} + \tau^2\rho_{k,k+1}\sigma_k\sigma_{k+1}F_kF_{k+1}\,\mathrm{d}t
\end{aligned}$$

and this can be re-expressed as:

$$\begin{aligned}
\mathrm{d}\phi = {} & D + \tau(1 + \tau F_{k+1})F_k\sigma_k\,\mathrm{d}W^{k+1}\\
& + \tau(1 + \tau F_k)F_{k+1}\sigma_{k+1}\,\mathrm{d}W^{k+1}
\end{aligned} \tag{7.2.64}$$

where the drift term D in Eq. (7.2.64) is given by:

$$D = \tau(1 + \tau F_{k+1})F_k\mu_k\,\mathrm{d}t + \tau^2\rho_{k,k+1}\sigma_k\sigma_{k+1}F_kF_{k+1}\,\mathrm{d}t \tag{7.2.65}$$

Now since ϕ is a martingale under $\mathbb{Q}^{k+1}$ we know that $D = 0$, and therefore Eq. (7.2.65) results in:

$$(1 + \tau F_{k+1}) F_k \mu_k \, \mathrm{d}t = -\rho_{k,k+1} \sigma_k \sigma_{k+1} F_k F_{k+1} \, \mathrm{d}t$$

This means that the drift is:

$$\mu_k = -\frac{\tau \rho_{k,k+1} \sigma_k \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} \tag{7.2.66}$$

Substituting for μ_k in Eq. (7.2.61) gives:

$$\mathrm{d}F_k = F_k \sigma_k \left\{ -\frac{\tau \rho_{k,k+1} \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} \, \mathrm{d}t + \mathrm{d}W^{k+1} \right\} \tag{7.2.67}$$

Comparing Eq. (7.2.67) with Eq. (7.2.40) we have:

$$\mathrm{d}W^k = -\frac{\tau \rho_{k,k+1} \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} \, \mathrm{d}t + \mathrm{d}W^{k+1} \tag{7.2.68}$$

which is the relationship between Brownian motions $\mathrm{d}W^k$ and $\mathrm{d}W^{k+1}$.

$i \geqslant k + 2$

Let the stochastic process followed by F_k under $\mathbb{Q}^{k+2}$ be:

$$\mathrm{d}F_k = F_k \mu_k \, \mathrm{d}t + F_k \sigma_k \, \mathrm{d}W^{k+2} \tag{7.2.69}$$

where W^{k+2} is Brownian motion under probability measure $\mathbb{Q}^{k+2}$, and drift term μ_k is to be found.

Replacing k with $k + 1$ in Eq. (7.2.68) gives:

$$\mathrm{d}W^{k+1} = -\frac{\tau \rho_{k+1,k+2} \sigma_{k+2} F_{k+2}}{1 + \tau F_{k+2}} \, \mathrm{d}t + \mathrm{d}W^{k+2} \tag{7.2.70}$$

and using Eq. (7.2.70) to substitute for $\mathrm{d}W^{k+1}$ in Eq. (7.2.68) gives:

$$\mathrm{d}W^k = -\frac{\tau \rho_{k,k+1} \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} \, \mathrm{d}t - \frac{\tau \rho_{k+1,k+2} \sigma_{k+2} F_{k+2}}{1 + \tau F_{k+2}} \, \mathrm{d}t + \mathrm{d}W^{k+2}$$

Substituting for $\mathrm{d}W^k$ in Eq. (7.2.40) gives:

$$\begin{aligned} \mathrm{d}W^k = &-F_k \sigma_k \left\{ \frac{\tau \rho_{k,k+1} \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} \, \mathrm{d}t + \frac{\tau \rho_{k+1,k+2} \sigma_{k+2} F_{k+2}}{1 + \tau F_{k+2}} \, \mathrm{d}t \right\} \\ &+ F_k \sigma_k \, \mathrm{d}W^{k+2} \end{aligned} \tag{7.2.71}$$

and thus the drift is a

$$\mu_k = -\frac{\tau \rho_{k,k+1} \sigma_{k+1} F_{k+1}}{1 + \tau F_{k+1}} - \frac{\tau \rho_{k+1,k+2} \sigma_{k+2} F_{k+2}}{1 + \tau F_{k+2}} \tag{7.2.72}$$

The following general expression can be derived in a similar manner:

$$\mathrm{d}F_k = -\sigma_k F_k \sum_{j=k+1}^{i} \frac{\rho_{k,j} \tau \sigma_j F_j}{1 + \tau F_j} \, \mathrm{d}t + \sigma_k F_k \, \mathrm{d}W^i \tag{7.2.73}$$

where all the symbols have the same meanings as before, but now i can take any integer value greater than k.

7.2.4 Interest rate quantos

This section considers derivatives whose value depends on the foreign interest rate yield curve but have a payoff in domestic currency. We use the same notation as in Section 7.3, which deals with foreign exchange derivatives.

For example, a standard interest rate caplet has a payoff in domestic currency and also depends on the domestic currency forward rates. The value at time t_0 of a caplet which pays at time t_k and extends from time t_{k-1} to time t_k is:

$$Caplet(t_0) = P\tau DF^d(t_0, t_k)E^{\mathbb{Q}^k}\left[\max\left(F_k^d - K, 0\right)\right] \tag{7.2.74}$$

where P is the principal, K is the strike, $\tau = t_{k-1} - t_k$, F_k is the domestic forward rate $F(t_0, t_{k-1}, t_k)$. Equation (7.2.74) can be evaluated using the Black–Scholes formula as follows:

$$Caplet(t_0) = P\tau DF^d(t_0, t_k)\left\{F^d(t_0, t_{k-1}, t_k)N_1(d_1) - KN_1(d_2)\right\} \tag{7.2.75}$$

where σ_d is the volatility of F_k, and

$$\begin{aligned} d_1 &= \frac{\log(F^d(t_0, t_{k-1}, t_k)/K) + \frac{\sigma_d^2}{2}(t_{k-1} - t_0)}{\sigma_d\sqrt{t_{k-1} - t_0}} \\ d_2 &= \frac{\log(F^d(t_0, t_{k-1}, t_k)/K) - \frac{\sigma_d^2}{2}(t_{k-1} - t_0)}{\sigma_d\sqrt{t_{k-1} - t_0}} \end{aligned} \tag{7.2.76}$$

In Section 7.2.3 we showed that the process $(F^d(t, t_{k-1}, t_k)/DF^d(t, t_k))$ is a martingale, that is has zero drift when $DF^d(t, t_k)$ is used as a numeraire.

$$\mathrm{d}\left(F^d(t, t_{k-1}, t_k)\right) = \sigma_d F^d(t, t_{k-1}, t_k)\,\mathrm{d}W_k^{\mathbb{Q}}$$

Quanto caplet

In a quanto caplet instead of using the domestic forward rate $F^d(t_0, t_{k-1}, t_k)$ we use the foreign forward rate $F^f(t_0, t_{k-1}, t_k)$. Under the probability measure $\mathbb{F}$ associated with a foreign zero coupon bond $DF^f(t, k_k)$ the foreign forward rate is a martingale, and is described by the following equation:

$$\mathrm{d}\left(F^f(t, t_{k-1}, t_k)\right) = \sigma_f F^f(t, t_{k-1}, t_k)\,\mathrm{d}W_f^{\mathbb{F}} \tag{7.2.77}$$

However, when we use $DF^d(t, t_k)$ as a numeraire the process $(F^f(t, t_{k-1}, t_k)/DF^d(t, t_k))$ has a drift, and follows the process:

$$\mathrm{d}\left(F^f(t, t_{k-1}, t_k)\right) = F^f(t, t_{k-1}, t_k)\alpha\,\mathrm{d}t + \sigma_f F^f(t, t_{k-1}, T_k)\,\mathrm{d}W_f^{\mathbb{Q}} \tag{7.2.78}$$

Our aim is to find the value of α and then price the quanto caplet using:

$$\begin{aligned} QCaplet(t_0) = {} & P\tau DF^d(t_0, t_k) \\ & \times \left\{F^f(t_0, t_{k-1}, t_k)\exp\left(\alpha(t_{k-1} - t_0)\right)N_1(d_1) - KN_1(d_2)\right\} \end{aligned} \tag{7.2.79}$$

where P is the principal, K is the strike, $\tau = t_{k-1} - t_k$, $F^f(t_0, t_{k-1}, t_k)$ is the foreign currency forward rate, σ_f is the volatility of $F^f(t_0, t_{k-1}, t_k)$ and

$$d_1 = \frac{\log(F^f(t_0, t_{k-1}, t_k)/K) + (\alpha + \sigma_f^2/2)(t_{k-1} - t_0)}{\sigma_f \sqrt{t_{k-1} - t_0}}$$
$$d_2 = \frac{\log(F^f(t_0, t_{k-1}, t_k)/K) + (\alpha - \sigma_f^2/2)(t_{k-1} - t_0)}{\sigma_f \sqrt{t_{k-1} - t_0}}$$

We will now derive the value of α.

First we define two processes $X_1(t)$ and $X_2(t)$ such that:

$$X_1(t) = \left(\frac{DF^f(t, t_{k-1}) - DF^f(t, t_k)}{DF^d(t, t_k)}\right) X_d^f(t) \tag{7.2.80}$$

and

$$X_2(t) = \tau X_d^f(t) \frac{DF^f(t, t_k)}{DF^d(t, t_k)} = \tau X_d^f(t, t_k) \tag{7.2.81}$$

where $X_d^f(t, t_k)$ is the forward foreign exchange rate (see Section 7.3). Therefore,

$$\frac{X_1(t)}{X_2(t)} = \frac{DF^f(t, t_{k-1}) - DF^f(t, t_k)}{\tau DF^f(t, t_k)} = F^f(t, t_{k-1}, t_k) \tag{7.2.82}$$

Now X_1 and X_2 are martingales under $\mathbb{Q}$ so we have:

$$\mathrm{d}X_1 = \sigma_1 X_1 \,\mathrm{d}W_1^{\mathbb{Q}} \tag{7.2.83}$$

and

$$\mathrm{d}X_2 = \sigma_2 X_2 \,\mathrm{d}W_2^{\mathbb{Q}} \tag{7.2.84}$$

Equation (7.2.84) can also be expressed as

$$\frac{\mathrm{d}(\tau X_d^f(t, t_k))}{\tau X_d^f(t, t_k)} = \sigma_x \,\mathrm{d}W_x^{\mathbb{Q}} \tag{7.2.85}$$

Using Ito we obtain:

$$\mathrm{d}\left(\frac{X_1}{X_2}\right) = \frac{X_1}{X_2}\{\sigma_1 \,\mathrm{d}W_1^{\mathbb{Q}} - \sigma_2 \,\mathrm{d}W_2^{\mathbb{Q}}\} + \frac{X_1}{X_2}\{E[(\sigma_2 \,\mathrm{d}W_2^{\mathbb{Q}})^2] - E[(\sigma_1 \,\mathrm{d}W_1^{\mathbb{Q}})(\sigma_2 \,\mathrm{d}W_2^{\mathbb{Q}})]\} \tag{7.2.86}$$

and the following processes for $\log(X_1)$ and $\log(X_2)$:

$$\mathrm{d}(\log(X_1)) = -\frac{\sigma_1^2}{2} \,\mathrm{d}t + \sigma_1 \,\mathrm{d}W_1^{\mathbb{Q}}$$
$$\mathrm{d}(\log(X_2)) = -\frac{\sigma_2^2}{2} \,\mathrm{d}t + \sigma_2 \,\mathrm{d}W_2^{\mathbb{Q}} \tag{7.2.87}$$

Using Eq. (7.2.85) we can write Eq. (7.2.87) as:

$$\mathrm{d}\big(\log\big(\tau X_d^f(t,t_k)\big)\big) = -\frac{\sigma_x^2}{2}\,\mathrm{d}t + \sigma_x\,\mathrm{d}W_x^{\mathbb{Q}} \tag{7.2.88}$$

Now,

$$\begin{aligned}
&E\big[\mathrm{d}\big(\log(X_2)\big)\big\{\mathrm{d}\big(\log(X_2)\big) - \mathrm{d}\big(\log(X_1)\big)\big\}\big] \\
&\quad = E\big[\big(\mathrm{d}\big(\log(X_2)\big)\big)^2\big] - E\big[\mathrm{d}\big(\log(X_1)\big)\,\mathrm{d}\big(\log(X_2)\big)\big] \\
&\quad = E\big[\big(\sigma_2\,\mathrm{d}W_2^{\mathbb{Q}}\big)^2\big] - E\big[\big(\sigma_1\,\mathrm{d}W_1^{\mathbb{Q}}\big)\big(\sigma_2\,\mathrm{d}W_2^{\mathbb{Q}}\big)\big]
\end{aligned}$$

where we have ignored terms in $\mathrm{d}t$ with order greater than 1.

In addition,

$$\begin{aligned}
&E\big[\mathrm{d}\big(\log(X_2)\big)\big\{\mathrm{d}\big(\log(X_2)\big) - \mathrm{d}\big(\log(X_1)\big)\big\}\big] \\
&\quad = -E\left[\mathrm{d}(\log(X_2))\,\mathrm{d}\left(\log\left(\frac{X_1}{X_2}\right)\right)\right] \\
&\quad = -E\big[\mathrm{d}\big(\log(X_2)\big)\,\mathrm{d}\big(\log\big(F^f(t,t_{k-1},t_k)\big)\big)\big] \\
&\quad = -E\big[\mathrm{d}\big(\log\big(\tau X_d^f(t,t_k)\big)\big)\,\mathrm{d}\big(\log\big(F^f(t,t_{k-1},t_k)\big)\big)\big] \\
&\quad = -E\big[\mathrm{d}\big(\log\big(X_d^f(t,t_k)\big)\big)\,\mathrm{d}\big(\log\big(F^f(t,t_{k-1},t_k)\big)\big)\big] \\
&\quad = -\sigma_x\sigma_f\rho_{x,f}\,\mathrm{d}t
\end{aligned}$$

where σ_x is the volatility of the forward foreign exchange rate $X_d^f(t,t_k)$, σ_f is the volatility of the foreign forward rate $F^f(t,t_{k-1},k)$, and $\rho_{x,f}$ is the correlation between $\mathrm{d}W_x^{\mathbb{Q}}$ and $\mathrm{d}W_f^{\mathbb{F}}$.

Therefore Eq. (7.2.86) can be written as:

$$\mathrm{d}\left(\frac{X_1}{X_2}\right) = -\frac{X_1}{X_2}\{\sigma_x\sigma_f\rho_{x,f}\}\,\mathrm{d}t + \frac{X_1}{X_2}\big\{\sigma_1\,\mathrm{d}W_1^{\mathbb{Q}} - \sigma_2\,\mathrm{d}W_2^{\mathbb{Q}}\big\}$$

which means that

$$\begin{aligned}
&\mathrm{d}\big(F^f(t,t_{k-1},t_k)\big) \\
&\quad = -F^f(t,t_{k-1},t_k)\{\sigma_x\sigma_f\rho_{x,f}\}\,\mathrm{d}t + F^f(t,t_{k-1},t_k)\big\{\sigma_1\,\mathrm{d}W_1^{\mathbb{Q}} - \sigma_2\,\mathrm{d}W_2^{\mathbb{Q}}\big\}
\end{aligned}$$

Comparing the above equation with Eq. (7.2.78):

$$\begin{aligned}
&\mathrm{d}\big(F^f(t,t_{k-1},t_k)\big) \\
&\quad = -F^f(t,t_{k-1},t_k)\{\sigma_x\sigma_f\rho_{x,F}\}\,\mathrm{d}t + \sigma_f F^f(t,t_{k-1},t_k)\,\mathrm{d}W^{\mathbb{Q}}
\end{aligned} \tag{7.2.89}$$

and so $\alpha = -\sigma_x\sigma_f\rho_{x,f}$.

Quanto floorlet

The formula to value a quanto floorlet can be obtained in similar manner to that used for the quanto caplet.

The value at time t_0 of a standard floorlet which pays at time t_k and extends from time t_{k-1} to time t_k is:

$$Floorlet(t_0) = P\tau DF^d(t_0, t_k)E^{\mathbb{Q}^k}\left[\max\left(K - F_k^d, 0\right)\right] \tag{7.2.90}$$

where P is the principal, K is the strike, $\tau = t_{k-1} - t_k$, F_k is the domestic forward rate $F(t_0, t_{k-1}, t_k)$. Equation (7.2.90) can be evaluated using the Black–Scholes formula as follows:

$$Floorlet(t_0) = P\tau DF^d(t_0, t_k)\left\{-F^d(t_0, t_{k-1}, t_k)N_1(-d_1) + KN_1(-d_2)\right\} \tag{7.2.91}$$

where the symbols have the same meaning as for the corresponding quanto caplet.

$$\begin{aligned} d_1 &= \frac{\log(F^d(t_0, t_{k-1}, t_k)/K) + \frac{\sigma_d^2}{2}(t_{k-1} - t_0)}{\sigma_d\sqrt{t_{k-1} - t_0}}, \\ d_2 &= \frac{\log(F^d(t_0, t_{k-1}, t_k/K) - \frac{\sigma_d^2}{2}(t_{k-1} - t_0)}{\sigma_d\sqrt{t_{k-1} - t_0}} \end{aligned} \tag{7.2.92}$$

In a quanto floorlet, instead of using the domestic forward rate $F^d(t_0, t_{k-1}, t_k)$, we use the foreign forward rate $F^f(t_0, t_{k-1}, t_k)$:

$$\begin{aligned} &QFloorlet(t_0) \\ &\quad = P\tau DF^d(t_0, t_k) \\ &\qquad \times \left\{-F^f(t_0, t_{k-1}, t_k)\exp\left(\alpha(t_{k-1} - t_0)\right)N_1(-d_1) + KN_1(-d_2)\right\} \end{aligned} \tag{7.2.93}$$

where P is the principal, K is the strike, $\tau = t_{k-1} - t_k$, $F^f(t_0, t_{k-1}, t_k)$ is the foreign currency forward rate, σ_f is the volatility of $F^f(t_0, t_{k-1}, t_k)$ and

$$\begin{aligned} d_1 &= \frac{\log(F^f(t_0, t_{k-1}, t_k)/K) + (\alpha + \frac{\sigma_f^2}{2})(t_{k-1} - t_0)}{\sigma_f\sqrt{t_{k-1} - t_0}} \\ d_2 &= \frac{\log(F^f(t_0, t_{k-1}, t_k)/K) + (\alpha - \frac{\sigma_f^2}{2})(t_{k-1} - t_0)}{\sigma_f\sqrt{t_{k-1} - t_0}} \end{aligned}$$

and as before $\alpha = -\sigma_x\sigma_f\rho_{x,f}$.

Quanto swaplet

A quanto (also known as *diff* or *differential*) swaplet is an agreement in which one party makes floating rate payments based on the foreign forward rate while the other makes fixed or floating payments based on the domestic interest rates.

Here we consider quanto swaplets in which the received floating leg coupons (in domestic currency) are computed using foreign forward rates.

The value of a standard swaplet (in which all the currencies are domestic) can be found by using Eqs. (7.2.90) and (7.2.74) to write:

$$\begin{aligned}
&Caplet(t_0) - Floorlet(t_0) \\
&\quad = P\tau DF^d(t_0,t_k)E^{\mathbb{Q}^k}\left[\max\left(F_k^d - K, 0\right) - \max\left(K - F_k^d, 0\right)\right] \\
&\quad = P\tau DF^d(t_0,t_k)E^{\mathbb{Q}^k}\left[F_k^d - K\right] \\
&\quad = P\tau DF^d(t_0,t_k)\left\{E^{\mathbb{Q}^k}\left[F_k^d\right] - K\right\} \\
&\quad = P\tau DF^d(t_0,t_k)\left\{F_k^d(t_0,t_{k-1},t_k) - K\right\} \qquad (7.2.94)
\end{aligned}$$

From Eq. (7.2.94) we can see that $Caplet(t_0) - Floorlet(t_0)$ is the value of a swaplet in which the owner pays the fixed rate K and receives the floating rate $F_k^d(t_0,t_{k-1},t_k)$.

The value of the floating leg payment can be found by setting $K = 0$ in Eq. (7.2.94), and is $P\tau DF^d(t_0,t_k)F_k^d(t_0,t_{k-1},t_k)$.

The value of the floating leg payments in a quanto swap can be found in a similar manner:

$$\begin{aligned}
&QCaplet(t_0) - QFloorlet(t_0) \\
&\quad = P\tau DF^d(t_0,t_k)E^{\mathbb{Q}^k}\left[\max\left(F_k^f - K, 0\right) - \max\left(K - F_k^f, 0\right)\right]
\end{aligned}$$

Substituting from Eqs. (7.2.79) and (7.2.93):

$$\begin{aligned}
&QCaplet(t_0) - QFloorlet(t_0) \\
&\quad = P\tau DF^d(t_0,t_k)\left\{F^f(t_0,t_{k-1},t_k)\exp\left(\alpha(t_{k-1}-t_0)\right)N_1(d_1) - KN_1(d_2)\right. \\
&\qquad\qquad \left. - F^f(t_0,t_{k-1},t_k)N_1(-d_1) + KN_1(-d_2)\right\}
\end{aligned}$$

When $K = 0$ we have $d_1 = \infty$, $N_1(d_1) = 1$ and $N_1(-d_1) = 0$, which means that the value of the floating leg payment is:

$$FloatLeg(t_0) = P\tau DF^d(t_0,t_k)F^f(t_0,t_{k-1},t_k)\exp\left(\alpha(t_{k-1}-t_0)\right) \qquad (7.2.95)$$

where as before $\alpha = -\sigma_x\sigma_f\rho_{x,f}$.

Using Eq. (7.2.95) the value of a quanto swaplet with (pay) fixed domestic rate and (receive) foreign floating rate is:

$$QSwaplet(t_0) = P\tau DF^d(t_0,t_k)\left\{F^f(t_0,t_{k-1},t_k)\exp\left(\alpha(t_{k-1}-t_0)\right) - K\right\}$$

7.3 Foreign exchange derivatives

Here we consider derivatives based on the exchange rate between a *domestic* currency and a *foreign* currency. We will use the convention that quantities relating to the domestic currency will have the superscript d, while those for the foreign currency will have the superscript f. The notation for the various exchange rates is as follows:

- $X_d^f(t)$ is the spot value of one unit of foreign currency in domestic currency at time t
- $X_b^f(t)$ is the spot value of one unit of foreign currency in base currency at time t
- $X_b^d(t)$ is the spot value of one unit of domestic currency in base currency at time t
- $X_d^f(t, T)$ is the (time t) forward value of one unit of foreign currency in domestic currency at time T.

Covered interest arbitrage

If the current spot exchange rate, $X_d^f(t)$, is known then using *covered interest arbitrage* it is possible to obtain a value for the future spot exchange rate—we denote this forward exchange rate by $X_d^f(t, T)$, where $T > t$.

Let us consider the following two scenarios:

Scenario A

At time t an investor deposits one unit of foreign currency which grows at the (constant) foreign risk free interest rate r^f. By time T the initial amount will have increased to $1/DF^f(t, T)$ units of foreign currency, where $DF^f(t, T) = \exp(-r^f(T - t))$. The foreign currency is then converted into domestic currency at the time T forward exchange rate $X_d^f(t, T)$, and thus yields $X_d^f(t, T)/DF^f(t, T)$ units of domestic currency.

Scenario B

At time t an investor deposits $X_d^f(t)$ units of domestic currency (the sum is *equivalent* to one unit of foreign currency), and this grows at the (constant) domestic risk free interest rate r^d. At time T the initial sum will have increased to $X_d^f(t)/DF^d(t, T)$ units of domestic currency, where $DF^d(t, T) = \exp(-r^d(T - t))$.

For no arbitrage to occur the final amount of domestic currency in both scenarios must be the same—we have assumed that there is no charge in converting one currency into another.

We thus have:

$$\frac{X_d^f(t, T)}{DF^f(t, T)} = \frac{X_d^f(t)}{DF^d(t, T)}$$

which means that the forward exchange rate, at time T is

$$X_d^f(t,T) = X_d^f(t)\frac{DF^f(t,T)}{DF^d(t,T)} \tag{7.3.1}$$

7.3.1 FX forward

An FX forward is a contract to exchange a given amount of domestic currency for an agreed amount of foreign currency at a future time T. If P^f is the amount (number of units) of foreign currency, and P^d is the amount (number of units) of domestic currency, then the value (in domestic currency) of the FX contract at time T is:

$$FX_d(T) = P^f X_d^f(T) - P^d$$

The value of the contract at time t is thus:

$$FX_d(t) = \left\{P^f X_d^f(t,T) - P^d\right\}DF^d(t,T)$$

where $t < T$. Substituting for $X_d^f(t,T)$ from Eq. (7.3.1) then gives:

$$FX_d(t) = \left\{P^f X_d^f(t)\frac{DF^f(t,T)}{DF^d(t,T)} - P^d\right\}DF^d(t,T)$$

which can be re-expressed as

$$FX_d(t) = P^f X_d^f(t)DF^f(t,T) - P^d DF^d(t,T) \tag{7.3.2}$$

The value of this FX forward contract in base currency is thus

$$FX_b(t) = \left\{P^f X_d^f(t)DF^f(t,T) - P^d DF^d(t,T)\right\}X_b^d$$

That is,

$$FX_b(t) = P^f X_b^f(t)DF^f(t,T) - P^d DF^d(t,T)X_b^d(t) \tag{7.3.3}$$

where we have used the fact that $X_d^f(t)X_b^d(t) = X_b^f(t)$.

An alternative way of expressing Eq. (7.3.3) is as:

$$FX_b(t) = P^f\left\{X_b^f(t)DF^f(t,T) - KDF^d(t,T)X_b^d(t)\right\} \tag{7.3.4}$$

where $K = P^d/P^f$. In the next section we will see that K, the agreed rate to be paid for one unit of foreign currency in units of domestic currency, corresponds to the *strike* of an FX call option.

7.3.2 European FX option

Foreign exchange options can be priced using the Black–Scholes formula (Garman and Kohlhagen, 1983). There are three processes involved in foreign ex-

change options and, under the real-world probability measure $\mathbb{P}$, they are:

$$\begin{aligned} \mathrm{d}B^f &= r^f B^f \,\mathrm{d}t \\ \mathrm{d}X_d^f &= X_d^f \mu \,\mathrm{d}t + X_d^f \sigma \,\mathrm{d}W^P, \quad \mathrm{d}W^P \sim \mathrm{N}(0,\, \mathrm{d}t) \\ \mathrm{d}B^d &= r^d B^d \,\mathrm{d}t \end{aligned} \tag{7.3.5}$$

where X_d^f is the value of one unit of foreign currency in units of domestic currency, B^d is the domestic money account where money grows at the (constant) risk free rate r^d, B^f is the foreign money account where money grows at the (constant) risk free rate r^f, and σ is the volatility of X_d^f. From a domestic point of view there are only two assets—the money market account B^d, and the value of the foreign money market account in domestic currency, $B^f X_d^f$.

From Ito's product rule in Chapter 2, and using Eq. (2.6.3) with $X_1 = X_d^f$ and $X_2 = B^d$, we have:

$$\mathrm{d}\left(X_d^f B^f\right) = X_d^f B^f \left\{r^f + \mu\right\} \mathrm{d}t + X_d^f B^f \sigma \,\mathrm{d}W^P$$

We will now choose B^d as the numeraire and obtain the process for $B^f X_d^f / B^d$ using the Ito quotient rule given in Chapter 2. Substituting $X_1 = B^f X_d^f$ and $X_2 = B^d$ in Eq. (2.6.8) we obtain:

$$\mathrm{d}\left(\frac{B^f X_d^f}{B^d}\right) = \frac{B^f X_d^f}{B^d}\left\{\left(r^f - r^d + \mu\right)\mathrm{d}t\right\} + \frac{B^f X_d^f}{B^d}\sigma \,\mathrm{d}W^P$$

If we choose the probability measure $\mathbb{Q}$ such that:

$$\mathrm{d}W^P = \mathrm{d}W^Q - \frac{\left(r^f - r^d + \mu\right)}{\sigma}\,\mathrm{d}t \tag{7.3.6}$$

then $(B^f X_d^f / B^d)$ is a martingale since

$$\mathrm{d}\left(\frac{B^f X_d^f}{B^d}\right) = \frac{B^f X_d^f}{B^d}\sigma \,\mathrm{d}W^Q$$

Substituting for $\mathrm{d}W^P$ in Eq. (7.3.5) yields:

$$\begin{aligned} \mathrm{d}X_d^f &= X_d^f \mu \,\mathrm{d}t + X_d^f \sigma \left\{\mathrm{d}W^Q - \frac{\left(r^f - r^d + \mu\right)\mathrm{d}t}{\sigma}\right\} \\ &= X_d^f \mu \,\mathrm{d}t - X_d^f \left(r^f - r^d + \mu\right)\mathrm{d}t + X_d^f \sigma \,\mathrm{d}W^Q \end{aligned}$$

so

$$\mathrm{d}X_d^f = X_d^f \left(r^d - r^f\right)\mathrm{d}t + X_d^f \sigma \,\mathrm{d}W^Q \tag{7.3.7}$$

It can be seen that Eq. (7.3.7) is identical to Eq. (4.4.53) if the following mapping is used:

$$S \to X_d^f, \qquad r \to r^d, \qquad q \to r^f \tag{7.3.8}$$

This means that the above mapping allows us to price European FX puts and calls with the Black–Scholes formulae given in Eqs. (4.4.55) and (4.4.56).

FX call

The time t value (in domestic currency) of an FX call to buy *one unit of foreign currency* can be found from Eq. (4.4.55) and the substitutions given in (7.3.8). We have:

$$C_d(t) = X_d^f(t)\exp\big(-r^f(T-t)\big)N(d_1) - K\exp\big(-r^d(T-t)\big)N(d_2) \quad (7.3.9)$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left\{\log\left(\frac{X_d^f(t)}{K}\right) + \big(r^d - r^f\big)(T-t) + \frac{1}{2}\sigma^2(T-t)\right\} \quad (7.3.10)$$

$$d_2 = d_1 - \sigma\sqrt{T-t} \quad (7.3.11)$$

where K is the strike the rate that has been agreed to pay for one unit of foreign currency in units of domestic currency, and σ is the implied foreign/domestic currency exchange rate FX option volatility, which may depend on effects such as time to maturity, volatility smile, etc.

In practice the following, modified version of Eq. (7.3.9) is usually used:

$$C_d(t) = P^f\left\{X_d^f(t)\exp\big(-r^f(T-t)\big)N(d_1) - K\exp\big(-r^d(T-t)\big)N(d_2)\right\} \quad (7.3.12)$$

where P^f is the number of units of foreign currency and all the other symbols have their previous meanings.

The value $C_b(t)$ of the call option in base currency can be found by using $C_b(t) = X_b^d C_d(t)$. From Eq. (7.3.12) we have:

$$C_b(t) = P^f\left\{X_b^f(t)N(d_1)DF^f(t,T) - KX_b^d(t)N(d_2)DF^d(t,T)\right\} \quad (7.3.13)$$

where we have used the fact that $X_b^d(t)X_d^f(t) = X_b^f(t)$, $DF^f(t,T) = \exp(-r^f(T-t))$ and $DF^d(t,T) = \exp(-r^d(T-t))$. We can also re-express the values for d_1 given in Eq. (7.3.10) as:

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left\{\log\left(\frac{X_b^f(t)}{KX_b^d(t)}\right) + \log\left(\frac{DF^f(t,T)}{DF^d(t,T)}\right) + \frac{1}{2}\sigma^2(T-t)\right\} \quad (7.3.14)$$

where we have used the fact that

$$\log\big(DF^f(t,T)\big) = -r^f(T-t), \qquad \log\big(DF^d(t,T)\big) = -r^d(T-t)$$

and

$$\log\left(\frac{X_d^f(t)}{K}\right) = \log\left(\frac{X_b^f(t)X_b^d(t)}{KX_b^d(t)}\right) = \log\left(\frac{X_b^f(t)}{KX_b^d(t)}\right)$$

We note that the term KX_b^d is the strike in units of base currency—that is, the amount that has been agreed to pay for one unit of foreign currency in units of *base currency*.

In the case when $N(d_1) = N(d_2) = 1$ (i.e., there is no uncertainty) Eq. (7.3.13) becomes:

$$C_b(t) = P^f \left\{ X_b^f(t) DF^f(t,T) - K X_b^d(t) DF^d(t,T) \right\}$$

which is the same as that already given in Eq. (7.3.4) for the FX forward.

FX put

The time t value of the corresponding put in units of base currency is:

$$P_b(t) = P^f \left\{ -X_b^f(t) DF^f(t,T) N(-d_1) + K X_b^d(t) DF^d(t,T) N(-d_2) \right\} \tag{7.3.15}$$

where the symbols have the same meanings as for the FX call.

7.4 Credit derivatives

Credit derivatives take into account the fact that a counterparty may not honor (for reasons of bankruptcy, etc.) the obligations set out in a given financial contract. In order to obtain the time t value of these derivatives it is necessary to determine the probability that the counterparty (and thus the contract) will survive until some future time $T > t$. Here we will denote the survival probability between times t and T by $S(t,T)$, and we compute its value from the hazard rate.

The hazard rate

As previously mentioned the survival probability between times t and T, where $T > t$, is denoted by $S(t,T)$. This means that the probability of default between times t and T is:

$$P_{\text{def}}(t,T) = 1 - S(t,T)$$

and the probability of default, as seen from time t, between times T_1 and T_2 is:

$$P_{\text{def}}(t,T_1,T_2) = S(t,T_1) - S(t,T_2), \quad T_2 > T_1 \tag{7.4.1}$$

The time t *discrete hazard rate* between times T and $T + \Delta T$, denoted by $H(t,T,T+\Delta T)$ is defined by:

$$\begin{aligned} H(t,T,T+\Delta T) &= \frac{1}{\Delta T} \frac{P_{\text{def}}(t,T,T+\Delta T)}{S(t,T+\Delta T)} \\ &= \frac{1}{\Delta T} \frac{S(t,T) - S(t,T+\Delta T)}{S(t,T+\Delta T)} \end{aligned}$$

which means that

$$H(t,T,T+\Delta T) = -\frac{S(t,T+\Delta T) - S(t,T)}{\Delta T} \frac{1}{S(t,T+\Delta T)} \tag{7.4.2}$$

As $\Delta T \to 0$, $H(t, T, T + \Delta T) \to h(t, T)$, where $h(t, T)$ is termed the *continuous hazard rate* between times t and T.

We observe that as $\Delta T \to 0$ Eq. (7.4.2) becomes:

$$h(t, T) = -\frac{\partial S(t, T)}{\partial T} \frac{1}{S(t, T)} \tag{7.4.3}$$

Now Eq. (7.4.3) can be re-expressed as:

$$h(t, s) = -\frac{\partial S(t, s)}{\partial s} \frac{\partial\{\ln(S(t, s))\}}{\partial S(t, s)} = -\frac{\partial\{\ln(S(t, s))\}}{\partial s} \tag{7.4.4}$$

where $s > t$, and we have used:

$$\frac{1}{S(t, T)} = -\frac{\partial\{\ln(S(t, T))\}}{\partial S(t, T)}$$

Integrating Eq. (7.4.4) yields:

$$\begin{aligned} \int_{s=t}^{T} h(t, s)\, \mathrm{d}s &= -\int_{s=t}^{T} \mathrm{d}\big\{\ln\big(S(t, s)\big)\big\} \\ &= -\big\{\ln\big(S(t, T)\big) - \ln\big(S(t, t)\big)\big\} \\ &= -\ln\big(S(t, T)\big) \end{aligned} \tag{7.4.5}$$

where we have used $S(t, t) = 1$ and $\ln(S(t, t)) = 0$.

So using Eq. (7.4.5) the survival probability can be expressed as:

$$S(t, T) = \exp\left\{-\int_{s=t}^{T} h(t, s)\, \mathrm{d}s\right\} \tag{7.4.6}$$

or

$$S(t, T) = \exp\big\{I(t, T)\big\} \tag{7.4.7}$$

where $I(t, T)$ is the cummulative hazard rate from time t to time T.

It is usual to approximate $I(t, T)$ as follows:

$$I(t, T) \sim I(t, t_k) = \sum_{i=1}^{k} h(t_{i-1}, t)(t_i - t_{i-1}) \tag{7.4.8}$$

where $t_0 = t$, $t_k = T$, and the following section gives details on how to estimate $h(t_{i-1}, t)$ from market *observables*.

One way of representing the hazard rates is to use a *hazard rate curve* which is defined as:

$$\big\{t_0, I(t_0, t_i)\big\}, \quad i = 0, , \ldots, n \tag{7.4.9}$$

If we further define $t_0 = t = 0$ and $t_n = T$, then Eq. (7.4.9) becomes:

$$\{0, 0\}, \big\{t_1, I(0, t_1)\big\}, \big\{t_2, I(0, t_2)\big\}, \ldots, \big\{t_n, I(0, t_n)\big\} \tag{7.4.10}$$

where we have used the fact that $I(t_0, t_0) = I(0, 0) = 0$.

Estimating the hazard rate from market observables

From Eq. (7.2.10) we know that the time t forward rate between times T_1 and T_2 is given by:

$$F(t, T_1, T_2) = \frac{1}{T_2 - T_1}\left\{\frac{DF(t, T_1)}{DF(t, T_2)} - 1\right\} \tag{7.4.11}$$

where $DF(t, T_1)$ and $DF(t, T_2)$ are the prices of the nondefaultable zero coupon bonds with maturities T_1 and T_2, respectively.

Letting $T_1 = T$ and $T_2 = T + \Delta T$ we obtain:

$$F(t, T, T + \Delta T) = \frac{1}{\Delta T}\left\{\frac{DF(t, T)}{DF(t, T + \Delta T)} - 1\right\} \tag{7.4.12}$$

which can be re-expressed as

$$\begin{aligned} &F(t, T, T + \Delta T) \\ &\quad = -\frac{DF(t, T + \Delta T) - DF(t, T)}{\Delta T}\frac{1}{DF(t, T + \Delta T)} \end{aligned} \tag{7.4.13}$$

If $\Delta T \to 0$ then $F(t, T, T + \Delta T) \to f(t, T)$ and from Eq. (7.4.13) we obtain:

$$\begin{aligned} f(t, T) &= -\frac{\partial DF(t, T)}{\partial T}\frac{1}{DF(t, T)} \\ &= -\frac{\partial DF(t, T)}{\partial T}\frac{\ln(DF(t, T))}{\partial DF(t, T)} \end{aligned} \tag{7.4.14}$$

Using Eq. (7.4.14) the instantaneous forward rate computed using nondefaultable zero coupon bond prices is:

$$f(t, T) = -\frac{\ln(DF(t, T))}{\partial T} \tag{7.4.15}$$

and the corresponding instantaneous forward rate computed from defaultable zero coupon bond prices is:

$$\bar{f}(t, T) = -\frac{\ln(\overline{DF}(t, T))}{\partial T} \tag{7.4.16}$$

Taking the survival probability $S(t, T)$ to be the ratio of the prices of defaultable and nondefaultable zero coupon bonds:

$$S(t, T) = \frac{\overline{DF}(t, T)}{DF(t, T)} \tag{7.4.17}$$

Now from Eqs. (7.4.15) and (7.4.16) we have:

$$\begin{aligned} \bar{f}(t, T) - f(t, T) &= -\frac{\ln(\overline{DF}(t, T) - DF(t, T))}{\partial T} \\ &= -\frac{\partial}{\partial T}\ln\left\{\frac{\overline{DF}(t, T)}{DF(t, T)}\right\} \end{aligned} \tag{7.4.18}$$

so from Eqs. (7.4.18) and (7.4.17):

$$\bar{f}(t,T) - f(t,T) = -\frac{\partial}{\partial T}\ln\big(S(t,T)\big) \tag{7.4.19}$$

Combining Eqs. (7.4.19) and (7.4.4) we have:

$$h(t,T) = \bar{f}(t,T) - f(t,T) \tag{7.4.20}$$

This means that we can compute the hazard rate $h(t,T)$ by taking the difference between the instantaneous forward rates computed using defaultable and nondefaultable zero coupon bonds.

7.4.1 Defaultable bond

For a defaultable bond we need to take into account the fact that the bond issuer may default, that is cease to make the bond coupon payments.

The time t value of a defaultable bond is:

$$\begin{aligned}\overline{B}(t) = {} & P\overline{DF}(t,t_m) && \text{principal}\\ & + \sum_{j=1}^{m} C_j \overline{DF}(t,t_j) && \text{coupons}\\ & + PR\sum_{j=1}^{m} DF(t,t_j)\big\{S(t,t_{j-1}) - S(t,t_j)\big\} && \text{recovery value}\end{aligned}$$

where t_m is the maturity of the bond, P is the principal, C_j is the value of the jth coupon, R is the recovery rate, $S(t,t_j)$ is the probability that the bond will survive until time t_j, and the zero coupon defaultable bond prices are defined by $\overline{DF}(t,t_m) = S(t,t_m)DF(t,t_m)$ and $\overline{DF}(t,t_j) = DF(t,t_j)S(t,t_j)$. The term $\{S(t,t_{j-1}) - S(t,t_j)\}$ is the probability that the bond will default between times t_{j-1} and t_j.

7.4.2 Credit default swap

A Credit Default Swap (CDS) is a contract between two counterparties in which one (say A) makes periodic fixed payments to the other (say B) in order to obtain protection on the default of a reference credit. In the event of default, B pays A the default payment of $1 - R$, where R is the recovery rate, and the contract ceases.

The time t value of the credit default swap to A, the purchaser of the insurance, is:

$$\begin{aligned}CDS(t) = {} & -\sum_{j=1}^{m} C_j \overline{DF}(t,t_j) && \text{coupons}\\ & + P(1-R)\sum_{j=1}^{m} DF(t,t_j)\big\{S(t,t_{j-1}) - S(t,t_j)\big\} && \text{recovery value}\end{aligned}$$

where the symbols have the same meanings as for the defaultable bond.

7.4.3 Total return swap

A Total Return Swap (TRS) is a synthetic replication of the return of a reference asset (bond) B. The *receiver* of the TRS receives the coupon payments of the reference asset during the life of the swap in return for making periodic coupon payments at the risk-free floating rate plus an agreed margin. In the event of default, the receiver makes a default payment to the payer equal to the agreed initial price of the reference asset less the price at default, and the transaction terminates. If there is no default then the difference between the initial asset (bond) price B_0 and the price at maturity $B(t_m)$ is settled between the payer and the receiver, with the receiver paying (receiving) if the asset (bond) is worth less (more) at maturity. The maturity of the reference asset (bond) may be longer than the maturity t_m of the swap.

To the receiver of the reference asset coupons a TRS has value TRS_r, which is given by:

$$TRS_r = \text{total bond return} - \text{total floating coupon payments of the swap}$$

where total bond return is given by:

> total bond return = total bond fixed coupons over the duration of the TRS+increase in the bond value at maturity of the TRS−default payment, if the bond defaults over the duration of the TRS

The value of TRS_r at time t is:

$$\begin{aligned}
&TRS_r(t)\\
&= \overline{C} \sum_{t_j=t_1}^{t_m} \overline{DF}(t,t_j) && \text{fixed reference bond payments}\\
&\quad + P\left\{\frac{B(t_m)-B_0}{B_0}\right\}\overline{DF}(t,t_m) && \text{increase in value of reference bond}\\
&\quad - \sum_{t_j=t_1}^{t_m} C_j \overline{DF}(t,t_j) && \text{floating payments of swap}\\
&&& \text{at LIBOR + margin}\\
&\quad - P(1-R)\sum_{t_j=t_1}^{t_m} DF(t,t_j)\\
&\quad \times \left\{S(t,t_{j-1}) - S(t,t_j)\right\} && \text{bond default payments}
\end{aligned}$$

and the reference bond satisfies:

$$B(t_R) = 1$$

$$B(t_m) = \overline{DF}(t_m,t_R)|t + \overline{C}\sum_{t_j=t_m}^{t_R} \overline{DF}(t_m,t_j)|t$$

$$+ R \sum_{t_j = t_m}^{t_R} DF(t_m, t_j)|t\{S(t, t_{j-1}) - S(t_m, t_j)\}$$

All symbols already defined in this chapter have their previous meanings. In addition,

P – the swap principal
$\overline{C}$ – the fixed coupon of the reference bond
t_m – the swap maturity
t_R – the maturity of the reference bond B
B_0 – the initial price of the reference bond
$B(t_m)$ – the final price of the reference bond (at swap maturity)
C_j – the floating coupon payment at time t_j. It is computed as:

$$C_j = P\{F(t, t_{j-1}, t_j) + margin^{TRS}\}\{t_j - t_{j-1}\}$$

$DF(t_1, t_2)|t$ – the discount factor between times t_1 and t_2 (as seen from time t) is:

$$DF(t_1, t_2)|t = \frac{D(t, t_2)}{D(t, t_1)}$$

$\overline{DF}(t_1, t_2)|t$ – the defaultable discount factor between times t_1 and t_2 (as seen from time t) is:

$$\overline{DF}(t_1, t_2)|t = \frac{\overline{DF}(t, t_2)}{\overline{DF}(t, t_1)} = \frac{DF(t, t_2)}{DF(t, t_1)} \frac{S(t, t_2)}{S(t, t_1)}$$

$\overline{DF}(t, t_1)$ – the defaultable discount factor between times t and t_1 is:

$$\overline{DF}(t, t_1) = DF(t, t_1)S(t, t_1)$$

7.5 Equity derivatives

7.5.1 Total return swap

An equity total return swap consists of an equity leg (whose coupons are determined by the change in value of the equity) and a floating leg which pays according to the forwards of the floating interest rate.

Here we ignore the effect of equity dividends and also assume that the currencies for both the floating and equity legs of the swap are the same.

Equity leg

Let the equity leg be specified by coupon payments at times $t_k, k = 1, \ldots, N_e$, where $\tau_e = t_k - t_{k-1}$. If, at time t the next coupon payment occurs at t_i then the value of the equity leg is:

$$V_e(t) = \left\{ \frac{S(t)}{DF(t,t_i)} - L(t_{i-1}) \right\} DF(t,t_i) + \sum_{k=i+1}^{N_e} \left\{ \frac{S(t)}{DF(t,t_k)} - \frac{S(t)}{DF(t,t_{k-1})} \right\} DF(t,t_k) \tag{7.5.1}$$

where $S(t)$ is the equity value at current time t, $L(t_{i-1})$ is the *reset* value of the equity at time t_{i-1}, and $DF(t,t_k)$ is the discount factor between times t and t_k, $t_k > t$. It can be seen that the value of the equity $S(t_1)$ at time $t_1 > t$ is obtained by inflating the current value, $S(t)$ by the reciprocal of the discount factor, $DF(t,t_1)$; that is, $S(t_1) = S(t)/DF(t,t_1)$.

Floating leg

Let the floating leg have coupon payments at times $t_m, m = 1, \ldots, N_f$, where $\tau_f = t_m - t_{m-1}$. If the next coupon is at time t_j, then the value of the floating leg is:

$$V_f(t) = L(t_{j-1})\{R(t_{j-1}) + \Phi\}\tau_f DF(t,t_j) + \sum_{m=j+1}^{N_f} \frac{S(t)}{DF(t,t_{m-1})} \{F(t,t_{m-1},t_m) + \Phi\}\tau_f DF(t,t_m) \tag{7.5.2}$$

where Φ is the *margin* added to the forward rate used to compute coupons, $F(t,t_{m-1},t_m)$ is the time t forward rate between times t_{m-1} and t_m, and $R(t_{j-1})$ is the reset rate that is used between times t_{j-1} and t_j to compute the coupon payment at time t_j.

Payer equity total return swap

The owner of a payer equity TRS pays the equity leg coupons, and thus at time t the swap has value:

$$ETRS_p(t) = V_f(t) - V_e(t) \tag{7.5.3}$$

The owner of a receiver equity TRS receives the equity leg coupons, and the value of the swap is:

$$ETRS_r(t) = -V_f(t) + V_e(t) \tag{7.5.4}$$

We will now compute an expression for the value of a payer equity swap. Since

$$F(t,t_{m-1},t_m) = \frac{1}{t_m - t_{m-1}} \left(\frac{DF(t,t_{m-1})}{DF(t,t_m)} - 1 \right)$$

we have:

$$\tau_f F(t,t_{m-1},t_m) = \frac{DF(t,t_{m-1})}{DF(t,t_m)} - 1 \tag{7.5.5}$$

Substituting Eq. (7.5.5) into Eq. (7.5.2) and using Eq. (7.5.3) we obtain:

$$
\begin{aligned}
ETRS_p(t) = & \sum_{m=j+1}^{N_f} \frac{S(t)DF(t,t_m)}{DF(t,t_{m-1})}\left\{\frac{DF(t,t_{m-1})}{DF(t,t_m)} - 1\right\} \\
& + \sum_{m=j+1}^{N_f} \frac{S(t)}{DF(t,t_{m-1})}\Phi\tau_f DF(t,t_m) \\
& + L(t_{j-1})(R(t_{j-1}) + \Phi)\tau_f DF(t,t_j) \\
& - \sum_{k=i+1}^{N_e} S(t)\left\{1 - \frac{DF(t,t_k)}{DF(t,t_{k-1})}\right\} \\
& - \left\{\frac{S(t)}{DF(t,t_i)} - L(t_{i-1})\right\}DF(t,t_i)
\end{aligned} \tag{7.5.6}
$$

If $N_e = N_f = N$, and $\tau_e = \tau_f = \tau$, then all the equity and float leg payments coincide and Eq. (7.5.6) simplifies to:

$$
\begin{aligned}
ETRS_p(t) = & S(t)\sum_{k=j+1}^{N} \frac{DF(t,t_k)}{DF(t,t_{k-1})}\Phi\tau DF(t,t_k) \\
& + L(R+\Phi)\tau DF(t,t_j) - \left\{\frac{S(t)}{DF(t,t_j)} - L\right\}DF(t,t_j)
\end{aligned} \tag{7.5.7}
$$

Thus, if the spread Φ is zero the value of the payer equity TRS is:

$$
\begin{aligned}
ETRS_p(t) = & L(t_{j-1})R(t_{j-1})\tau DF(t,t_j) \\
& - \left\{\frac{S(t)}{DF(t,t_j)} - L(t_{j-1})\right\}DF(t,t_j)
\end{aligned} \tag{7.5.8}
$$

In these circumstances the value of the equity TRS at time t only depends on the current swaplet, which extends from t_{j-1} to t_j, where $t_{j-1} < t < t_j$.

Equity swap

A special case of an equity TRS is an equity swap. Here one party (say A) pays the total returns on a given equity and receives (from party B) the returns on another equity, together with the interest on the net difference of the last reset notional of the two equity assets. An equity swap, *ESWP*, can be constructed from a *structured deal* consisting of a long position in one equity TRS and a short position in another equity TRS, with the same coupon payment dates and currency. If the individual equity TRS deals are denoted by $ETRS_p^1$ and $ETRS_p^2$, then the value of the equity swap at time t is:

$$
ESWP(t) = ETRS_p^1(t) - ETRS_p^2(t) \tag{7.5.9}
$$

Substituting Eq. (7.5.8) into Eq. (7.5.9) we have:

$$ESWP(t) = L^1(t_{j-1})R(t_{j-1})\tau DF(t,t_j) - \left\{\frac{S^1(t)}{DF(t,t_j)} - L^1(t_{j-1})\right\}DF(t,t_j)$$
$$- \left\{L^2(t_{j-1})R(t_{j-1})\tau DF(t,t_j)\right.$$
$$\left. - \left\{\frac{S^2(t)}{DF(t,t_j)} - L^2(t_{j-1})\right\}DF(t,t_j)\right\}$$

so the value of the equity swap to party A is:

$$ESWP(t) = -\left\{\frac{S^1(t)}{DF(t,t_j)} - L^1(t_{j-1})\right\}DF(t,t_j)$$
equity 1 returns paid by A
$$+ \left(L^1(t_{j-1}) - L^2(t_{j-1})\right)R(t_{j-1})\tau DF(t,t_j)$$
interest on difference of reset notionals paid by B
$$+ \left\{\frac{S^2(t)}{DF(t,t_j)} - L^2(t_{j-1})\right\}DF(t,t_j)$$
equity 2 returns paid by B

7.5.2 Equity quantos

The Black–Scholes equation can also be used to price equity quanto options (Reiner (1992)). We have the following processes:

$$\begin{aligned} \mathrm{d}S^f &= \mu_s S^f \,\mathrm{d}t + \sigma_s S^f \,\mathrm{d}W_s^P \\ \mathrm{d}X_d^f &= \mu_x X_d^f \,\mathrm{d}t + \sigma_x X_d^f \,\mathrm{d}W_x^P \\ \mathrm{d}B^f &= r^f B^f \,\mathrm{d}t \\ \mathrm{d}B^d &= r^d B^d \,\mathrm{d}t \end{aligned} \tag{7.5.10}$$

Here S^f is the price (in foreign currency units) of the foreign stock. B^d is the domestic money market account where money grows at the (constant) risk free interest rate r^d. B^f is the foreign money market account where money grows at the (constant) risk free interest rate r^f. X_d^f is the foreign exchange rate, that is the value of one unit of foreign currency in units of domestic currency.

The tradables for the domestic investor are the foreign money market account priced in domestic currency units (that is, $X_d^f B^f$) and the foreign stock priced in domestic currency units, $X_d^f S^f$.

We know from Eq. (4.4.30) that there is a probability measure (the risk neutral measure) $\mathbb{Q}$ under which the relative price of domestic tradables such as equities are martingales. Also we established that under $\mathbb{Q}$ the process followed by these tradables is GBM with constant drift r^d. So the process for the domestic equity S^d is:

$$\mathrm{d}S^d = S^d r^d \,\mathrm{d}t + \sigma S^d \,\mathrm{d}W^{\mathbb{Q}} \tag{7.5.11}$$

Similarly the process followed by the price of a foreign equity S^f under the foreign risk neutral measure $\mathbb{F}$ is:

$$\mathrm{d}S^f = S^f r^f \,\mathrm{d}t + \sigma_s S^f \,\mathrm{d}W_s^{\mathbb{F}} \tag{7.5.12}$$

However, the process followed by the price of a foreign equity S^f under the domestic risk neutal measure $\mathbb{Q}$ is:

$$\mathrm{d}S^f = S^f\left(r^f + \alpha\right)\mathrm{d}t + \sigma_s S^f \,\mathrm{d}W_s^{\mathbb{Q}} \tag{7.5.13}$$

where α (the *quanto adjustment*) is to be determined.

We will now derive the value for α, and then use this to price both quanto forwards and quanto options.

Determining the quanto adjustment, α

Since $X_d^f B^f$ and $X_d^f S^f$ are domestic tradables it means that the relative prices $X_d^f B^f/B^d$ and $X_d^f S^f/B^d$ are also martingales under the probability measure $\mathbb{Q}$

Now since $X_d^f B^f/B^d$ is a martingale,

$$\mathrm{d}\left(\frac{X_d^f B^f}{B^d}\right) = \sigma_x \frac{X_d^f B^f}{B^d}\,\mathrm{d}W_x^Q \tag{7.5.14}$$

We will start by writing $X_d^f B^f/B^d$ as $(S^f/B^f)(X_d^f B^f/B^d)$. Using the Ito product rule we have:

$$\begin{aligned}&\mathrm{d}\left\{\frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\right\}\\ &\quad= \frac{X_d^f B^f}{B^d}\,\mathrm{d}\left(\frac{S^f}{B^f}\right) + \frac{S^f}{B^f}\,\mathrm{d}\left(\frac{X_d^f B^f}{B^d}\right) + E\left[\mathrm{d}\left(\frac{S^f}{B^f}\right)\mathrm{d}\left(\frac{X_d^f B^f}{B^d}\right)\right]\end{aligned} \tag{7.5.15}$$

Substituting for $\mathrm{d}(X_d^f B^f/B^d)$ from Eq. (7.5.14) into Eq. (7.5.15) gives:

$$\begin{aligned}\mathrm{d}\left\{\frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\right\} &= \frac{X_d^f B^f}{B^d}\,\mathrm{d}\left(\frac{S^f}{B^f}\right) + \sigma_x \frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\,\mathrm{d}W_x^Q\\ &\quad + E\left[\mathrm{d}\left(\frac{S^f}{B^f}\right)\mathrm{d}\left(\frac{X_d^f B^d}{B^f}\right)\right]\end{aligned} \tag{7.5.16}$$

Using the Ito quotient rule (see Section 1.7.2) with one source of randomness, Eqs. (7.5.13) and (7.5.10) yield:

$$\mathrm{d}\left(\frac{S^f}{B^f}\right) = \frac{S^f}{B^f}\alpha\,\mathrm{d}t + \frac{S^f}{B^f}\sigma_s\,\mathrm{d}W_s^Q \tag{7.5.17}$$

We now consider the term $E[\mathrm{d}(S^f/B^f)\,\mathrm{d}(X_d^f B^f/B^d)]$ in Eq. (7.5.16):

$$E\left[\mathrm{d}\left(\frac{S^f}{B^f}\right)\mathrm{d}\left(\frac{X_d^f B^f}{B^d}\right)\right]$$
$$= E\left[\frac{S^f}{B^f}\left\{\alpha\,\mathrm{d}t + \frac{S^f}{B^f}\sigma_s\,\mathrm{d}W_s^Q\right\}\sigma_x\frac{X_d^f B^f}{B^d}\,\mathrm{d}W_x^Q\right]$$
$$= \frac{X_d^f B^f}{B^d}\frac{S^f}{B^f}\alpha\,\mathrm{d}t\,\sigma_x E\left[\mathrm{d}W_x^Q\right] + \frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\sigma_s\sigma_x E\left[\mathrm{d}W_x^Q\,\mathrm{d}W_s^Q\right]$$

Since $E[\mathrm{d}W_x^Q] = 0$ and $E[\mathrm{d}W_x^Q\,\mathrm{d}W_s^Q] = \rho_{xs}\,\mathrm{d}t$

$$E\left[\mathrm{d}\left(\frac{S^f}{B^f}\right)\mathrm{d}\left(\frac{X_d^f B^d}{B^f}\right)\right] = \frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\sigma_s\sigma_x\rho_{xs}\,\mathrm{d}t \tag{7.5.18}$$

Using the values of $\mathrm{d}(S^f/B^f)$ and $E[\mathrm{d}(S^f/B^f)\,\mathrm{d}(X_d^f B^d/B^f)]$ from Eqs. (7.5.17) and (7.5.18) in Eq. (7.5.16) results in:

$$\mathrm{d}\left\{\frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\right\} = \frac{X_d^f B^f}{B^d}\frac{S^f}{B^f}\alpha\,\mathrm{d}t + \frac{X_d^f B^f}{B^d}\frac{S^f}{B^f}\sigma_s\sigma_x\,\mathrm{d}W_s^Q$$
$$+ \sigma_x\frac{X_d^f S^f}{B^d}\,\mathrm{d}W_x^Q + \frac{S^f}{B^f}\frac{X_d^f B^f}{B^d}\sigma_s\sigma_x\rho_{xs}\,\mathrm{d}t$$

Rearranging we obtain:

$$\mathrm{d}\left\{\frac{X_d^f S^f}{B^d}\right\} = \frac{X_d^f S^f}{B^d}\{\alpha + \sigma_s\sigma_x\rho_{xs}\}\,\mathrm{d}t$$
$$+ \frac{X_d^f S^f}{B^d}\left(\sigma_s\,\mathrm{d}W_s^Q + \sigma_x\,\mathrm{d}W_x^Q\right) \tag{7.5.19}$$

We already mentioned that $X_d^f B^f/B^d$ is a martingale under probability measure $\mathbb{Q}$ so the drift term in Eq. (7.5.19) must be zero. This means that:

$$\alpha = -\sigma_s\sigma_x\rho_{xs}$$

where σ_s is volatility of the foreign equity, σ_x is volatility of the foreign exchange rate and ρ_{xs} is the correlation between $\mathrm{d}W_s$ and $\mathrm{d}W_x$.

Equation (7.5.13) can then be written as:

$$\mathrm{d}S^f = \left\{r^f - \sigma_s\sigma_x\rho_{xs}\right\}S^f\,\mathrm{d}t + \sigma_s S^f\,\mathrm{d}W_s^Q \tag{7.5.20}$$

Equity quanto forward

The (time t) value of a domestic equity forward contract with maturity T is:

$$F(t) = DF^d(t,T)\left\{S^d(t,T) - K^d\right\}$$

where K^d is the strike in domestic currency and $S^d(t,T)$ is the domestic forward price.

To value an equity quanto forward contract we need to know the forward price of the foreign equity S^f.

It can be seen from Eqs. (4.4.31), (4.4.32) and (7.5.20) that this forward price is:

$$S^f(t,T) = S^f(t)\exp\big(\big(r^f - \sigma_x\sigma_s\rho_{xs}\big)(T-t)\big) \tag{7.5.21}$$

where $S^f(t)$ is the current price of the foreign equity and T is the maturity of the forward.

In an equity quanto forward the payoff is in foreign currency but it is converted to domestic currency at a predetermined exchange rate (which we denote here by X). The value equity quanto forward is thus:

$$QF(t) = DF^d(t,T)\big\{S^f(t,T) - K^f\big\}X$$

where X is the prespecified exchange rate, K^f is the strike in units of foreign currency, and $S^f(t,T)$ is the foreign forward equity price.

Equity quanto option

In Chapter 4 Eqs. (4.4.58) and (4.4.59) expressed the value of vanilla European put and call options as:

$$\begin{aligned}
Call(t) &= \exp\big(-r^d(T-t)\big)\big\{S^d(t,T)N_1(d_1) - EN_1(d_2)\big\}\\
Put(t) &= \exp\big(-r^d(T-t)\big)\big\{-S^d(t,T)N_1(-d_1) + EN_1(-d_2)\big\}\\
d_1 &= \frac{\log(S^d(t,T)/E^d) + (\sigma^2/2)\tau}{\sigma\sqrt{(T-t)}}\\
d_2 &= \frac{\log(S^d(t,T)/E^d) - (\sigma^2/2)(T-t)}{\sigma\sqrt{(T-t)}}
\end{aligned}$$

where we have used superscripts to denote the domestic currency, and the current equity forward price with maturity T is:

$$S^d(t,T) = S^d(t)\exp\big(r^d(T-t)\big), \quad t \leqslant T$$

The value of an equity quanto option can be found by substituting S^f for S^d in the above expression. We obtain:

$$\begin{aligned}
QCall(t) &= \exp\big(-r^d(T-t)\big)\big\{S^f(t,T)N_1(d_1) - E^fN_1(d_2)\big\}X\\
QPut(t) &= \exp\big(-r^d(T-t)\big)\big\{-S^f(t,T)N_1(-d_1) + E^fN_1(-d_2)\big\}X\\
d_1 &= \frac{\log(S^f(t,T)/E^f) + (\sigma_s^2/2)\tau}{\sigma_s\sqrt{(T-t)}}\\
d_2 &= \frac{\log(S^f(t,T)/E^f) - (\sigma_s^2/2)(T-t)}{\sigma_s\sqrt{(T-t)}}
\end{aligned}$$

where E^f is the strike in foreign currency units, $S^f(t,T)$ is the foreign equity forward price (obtained from Eq. (7.5.21)) and X is the prespecified exchange rate (number of units of domestic currency per foreign currency unit).

This page intentionally left blank

8 C# portfolio pricing application

8.1 Introduction

This section provides details concerning a C# application, created using Microsoft Visual Studio 2005 (version 2.0), which values the deals contained in a set of user specified portfolios. It has been included to illustrate how the pricing functions discussed in the previous sections of the book can be incorporated into the kind of software that may be found in a bank, financial institution, or educational establishment. Here we provide code excerpts from the application; however, additional source code is available from the publisher's website.

The main features of this application are as follows:

- User defined portfolios of deals can be valued without the need to compile either C or C# programs. This means that the application is easy to use and is accessible to those who may possess business or financial knowledge, but do not have the technical skills required to write computer code.
- This application only deals with equity, foreign exchange derivatives—the exception is the inclusion of an interest rate forward rate trade. However, the software can be easily extended to include new deal types.
- All input/output to the application is by means of text files which can be easily edited.
- The software is modular and consists of a main C# program which calls both a compiled C# `DealLibrary` and also a compiled C `Analytics_MathLib`. It should be mentioned that in many cases the C# deal class calls the C pricing function with *reasonable* values for various parameters. For example, the number of time steps in a binomial lattice is set internally and cannot be altered by the user. Also the initial random seed for all Monte Carlo simulation is set internally to the same number (111) and this cannot be changed by the user.

We will now provide a brief overview of each component part of the application; more detail will be given later when specific deal classes are described.

8.1.1 The C# code

The application is defined by the C# solution `PortfolioValuer.sln`, and the projects `DealLibrary.csproj` and `PortfolioValuer.csproj`.

The project `PortfolioValuer.csproj` is the driver for the application and contains the C# code given in Code excerpt 8.1. The user interface is simply a windows form with a button to start the application, and if required this could easily be customized by the reader. Once the application has been started, it first loads a market data file and then reads a portfolio configuration file to determine which portfolios are to be valued. Valuation then proceeds for all the deals contained in the portfolio files and the results are written to the appropriate portfolio results file.

The project `DealLibrary.csproj` is concerned with the valuation of the available trades in the application. A separate C# *deal class*, derived from a single abstract base class `BaseDeal`, is provided for each trade type. The `BaseDeal` class provides abstract methods such as `Price()` and `Validate()`, which need to be implemented in deal classes. We will be primarily concerned with the method `Price()`, which is used to return the current value of a trade. Deal valuation may either be performed completely in C# code or by calling appropriate C routines in the `Analytics_MathLib` dynamic link library—this contains C pricing functions discussed in previous sections of the book and also utilities (such as random number generators), etc.

Code excerpt 8.2 provides the declaration of `BaseDeal` and illustrates how the deal class `EquityOptionDeal` implements the necessary methods; the complete C# code for a range of deals is provided later.

It can be seen that the deal class needs to specify the attributes which it will present to the user, and in addition access market data such as the equity price.

8.1.2 The text files

The application is driven by a *portfolio configuration* file and also a number of *portfolio definition* files, each of which is in plain text format and thus can easily be edited by the user. The portfolio configuration file specifies the names of the portfolio files that are to be valued, and each portfolio definition file provides the details of the trades contained in a given portfolio. In addition there is a *market data* file (also in text format) which provides the market data required to value the trades.

A portfolio configuration file and also two portfolio definition files are given in Exhibit 8.1. It can be seen that comments may be included in the portfolio definition files; these are useful for documenting the deals contained in the portfolio. It is also possible to ignore a single trade by using a // at the beginning of a line. Complete sections of a portfolio definition file can also be ignored by using the C style comment syntax /**/.

The syntax of each deal entry in the Portfolio Definition file is as follows:

```
Trade=<DealClass>,Reference=<Descriptive Text>,DealClassParam1=<Value1>, . . . ,_
  DealClassParamN=<ValueN>
```

Each deal line must begin with an assignment to the `Trade` *attribute* using the syntax `Trade=<DealClass>`, where `<DealClass>` is the name of the C# class used to represent the given deal—i.e., `EquityOptionDeal` for an

```
namespace PortfolioValuer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            StreamReader sr_config;
            StreamReader sr_tests;
            StreamWriter sw;

            MessageBox.Show("Computing value of portfolios");
            string AppDir = Path.GetDirectoryName(Application.ExecutablePath);
            string config_filename = AppDir + "\\Portfolios.txt";
            string MD_filename = AppDir + "\\MarketData.txt";
            try {
                BaseDeal.LoadMarketData(MD_filename);
             // Load and execute the specified tests
                sr_config = new StreamReader(config_filename);
                string assembly_name = AppDir + "\\DealLibrary.dll";
                Assembly Assm = Assembly.LoadFrom(assembly_name);
                Type[] LoadedTypes = Assm.GetTypes();
                string test_file;
                int block_comment_depth = 0;
                double port_val = 0.0;
                string BaseCurrency = BaseDeal.GetBaseCurrency();
                while ((test_file = sr_config.ReadLine()) != null)
                {
                    sr_tests = new StreamReader(test_file + ".txt");
                    string current_test;
                    string sdate = DateTime.Now.ToString();
                    string results_filename = AppDir + "\\" + test_file + "results.txt";
                    bool append = false;
                    sw = new StreamWriter(results_filename, append, Encoding.ASCII);
                    sw.WriteLine("============================");
                    sw.WriteLine(test_file + " in units of " + BaseCurrency);
                    sw.WriteLine(test_file +"   :" +sdate);
                    sw.WriteLine("============================");
                    port_val = 0.0;
                    block_comment_depth = 0;
                    while ((current_test = sr_tests.ReadLine()) != null) {

                        block_comment_depth += ((current_test.Length > 1)_
                           && (current_test.Substring(0,2) == "/*")) ? 1:0;
                        block_comment_depth -= ((current_test.Length > 1)_
                           && (current_test.Substring(0, 2) == "*/")) ? 1 : 0;
                                          .   .   .
                         try
                         {
                            deal_value = ac1.Price(); // return the value of the deal
                         }
                         catch (Exception ex)
                         {
                            sr_config.Close();
                            sw.Close();
                            sr_tests.Close();
                            throw new Exception(ex.Message);
                         }
                         port_val +=  deal_value; // add to the value to the portfolio
                         string str_deal_val = deal_value.ToString(" 0.0000;-0.0000; 0.0000");
                         sw.WriteLine(str_deal_val + "=" + ac1.Reference+","+ac1.Name());
                      }

                    } // end of tests for a given portfolio
```

Code excerpt 8.1 The main driver for the C# portfolio valuer application. After reading the market data file, it prices the trades contained in the portfolio definition files. The total value of each portfolio is also computed.

```
public abstract class BaseDeal    // The declaration for BaseDeal
{
    public abstract string Name();

    public abstract double Price();
    protected abstract void Validate();
    public string Reference { get { return Reference_; } set { Reference_ = value; } }

    protected string Reference_;

    public string BaseCurrency {get { return MarketDataDictionaries.GetBaseCurrency(); }}
}


public class EquityOptionDeal: BaseDeal     // The declaration for EquityOptionDeal
{
    public string Equity { get { return EquityName_; } set { EquityName_ = value; } }
    public int NumberOfUnits { get { return NumberOfUnits_; }_
       set { NumberOfUnits_ = value; } }
    public double Time_To_Expiry { get { return Time_To_Expiry_; }_
       set { Time_To_Expiry_ = value; } }

                    .   .    .

 public override string Name()   // implement method Name()
 {
    return "Equity Option";
 }

 public override double Price()   // implement method Price()
 {
    Validate();

    double val=0.0;
    double[] greeks = new double[6];
    double s0 = 0.0;
    double fx_spot = 0.0;
    try
    {
       s0 = MarketDataDictionaries.EquityTable[EquityName_].Spot;_
          // get current equity price
       Currency_ = MarketDataDictionaries.EquityTable[EquityName_].Currency;_
          // get equity volatility (assumed constant)
       DividendYield_ = MarketDataDictionaries.EquityTable[EquityName_].DivYield;_
          // get equity dividend yield
    }
    catch
    {
      throw new Exception(Pre_string_ + "--- No Market Data supplied for " + EquityName_);
    }
                .     .      .

    val *= fx_spot * NumberOfUnits_; // return value in base currency
    return val;
  }

  protected override void Validate()   // implelement method Validate()
 {
     Pre_string_ = Name() + "  (" + Reference_ +")";
     if (Time_To_Expiry_ < 0.0)
     {
        throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than zero
                                                                                years");
     }
     if (Strike_ < 0.0)
     {
        throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
     }

             .    .    .
 }
```

Code excerpt 8.2 C# code showing the declaration of class BaseDeal and illustrating the implementation of methods Price(), Validate(), and Name().

```
                    // write the portfolio value
                    sw.WriteLine("========================");
                    string str_port_val = port_val.ToString(" 0.00;-0.00; 0.00");
                    sw.WriteLine("TOTAL VALUE = " + str_port_val + " " + BaseCurrency + "  ");
                    sw.WriteLine("========================");
                    sw.Close();
                    sr_tests.Close();
                }
                MessageBox.Show("Have completed computing portfolio values");
                sr_config.Close();
                // Clear the dictionaries
                BaseDeal.CurrencyTable.Clear();
                BaseDeal.EquityTable.Clear();
                BaseDeal.BondTable.Clear();
                BaseDeal.CorrelationTable.Clear();
               }
            }
            catch (Exception ex)
            {
               MessageBox.Show("Computation aborted : exception : " + ex.Message);
            }
        }
    }
```

Code excerpt 8.2 (*Continued*).

equity option. The other deal attributes are the public properties of `DealClass` and can be in any order. Deal valuation proceeds by first constructing an empty class object of type `DealClass`. The application then converts the string representation of the deal attribute values to the types expected by the `DealClass`, and assignment to the deal object occurs. Once the deal object has been populated with the required trade data, the deal's `Price()` method is run. This retrieves the required market data, computes the deal value, and returns this to the application for output to the portfolio results file. All deals have the Reference attribute which is used for the purposes of identification. The deal reference is a description (or alphanumeric code) assigned to the trade, for instance `Reference=EQ:LaserComm-1001` or `Reference=Tech-10008_Generic`. The portfolio results files contain both the value and reference for each deal which has been valued.

The portfolio driver file

```
EQ-Investments
Broad-Investments
```

Portfolio definition file: EQ-investments

```
// EQ-Investments : Only contains equities
//=======================================
Trade=EquityOptionDeal,Reference=EQ:LaserComm-1001,Equity=LaserComm,Volatility=0.1,_
Strike=95.0,Time_To_Expiry=1.5,OptionType=Put,ExerciseStyle=European

Trade=EquityOptionDeal,Reference=EQ:WebComm-1004A,Equity=WebComm,Volatility=0.1,_
Strike=95.0,Time_To_Expiry=1.0,OptionType=Call,ExerciseStyle=European

Trade=EquityOptionDeal,Reference=EQ:LaserComm-1004,Equity=LaserComm,Volatility=0.1,_
Strike=95.0,Time_To_Expiry=1.0,OptionType=Call,ExerciseStyle=American
```

```
Trade=TwoEquityOptionDeal,Reference=EQ:CompuKalc-1005,Equity1=Mobile-Tech,_
Equity2=CompuKalc,Volatility1=0.2,Volatility2=0.2,RiskFreeRate=0.1,Strike=44.0,_
Time_To_Expiry=0.8,OptionType=Call,MinMax=Minimum,ExerciseStyle=European

Trade=TwoEquityOptionDeal,Reference=EQ:MobileTech|CompuKalc-1006,Equity1=Mobile-Tech,_
Equity2=CompuKalc,Volatility1=0.2,Volatility2=0.2,RiskFreeRate=0.1,Strike=94.0,_
Time_To_Expiry=0.8,OptionType=Call,MinMax=Maximum,ExerciseStyle=European

Trade=ThreeEquityOptionDeal,Reference=Tech-10001,Equity1=LaserComm,Equity2=TelComm,_
Equity3=SmartWeb,Volatility1=0.2,Volatility2=0.2,Volatility3=0.2,NumberOfUnits=100,_
Strike=100.0,Time_To_Expiry=1.0,OptionType=Put,MinMax=Maximum,MonteCarlo=No

Trade=GenericEquityBasketOptionDeal,Reference=Tech-10008_Generic,Volatilities=0.2%0.2%0.2,_
Equities=LaserComm%TelComm%SmartWeb,NumberOfUnits=100,_
Strike=100.0,Time_To_Expiry=1.0,OptionType=Call,MinMax=Maximum,MonteCarlo=Yes

Trade=FourEquityOptionDeal,Reference=Drinks-20001,Equity1=Drinks-4U,Equity2=Beverage-Ltd,_
Equity3=H2O-Ltd,Equity4=Fine-Wines-Ltd,Volatility1=0.2,Volatility2=0.2,Volatility3=0.2,_
Volatility4=0.2,NumberOfUnits=100,Strike=100.0,_
Time_To_Expiry=1.0,OptionType=Call,MinMax=Maximum,MonteCarlo=No

Trade=DownOutEquityOptionDeal,Reference=JPCA_111,Equity=H2O-Ltd,Volatility=0.2,Strike=100.0,_
Time_To_Expiry=1.0,Barrier_Level=90.0,OptionType=Call

Trade=DoubleKnockOutCallEquityOptionDeal,Reference=JPCAPP_115,Equity=LaserComm,Volatility=0.2,
Strike=100.0,Time_To_Expiry=1.0,Lower_Barrier_Level=90.0,Upper_Barrier_Level=340.0
```

Portfolio definition file: broad-investments

In the C derivative pricing functions developed in the earlier part of this book, all the deal information such as asset price, risk free interest rate, etc. was passed explicitly to the pricing function.

For instance, let us consider the pricing of the simple `EquityOption`, `Reference=EQ:WebComm-1004A`, which is specified in Exhibit 8.1. The entry (which we will refer to as E_1 in the portfolio definition file EQ-investments) is:

```
Trade=EquityOptionDeal,Reference=EQ:WebComm-1004A,Equity=WebComm,Volatility=0.1,_
Strike=95.0,Time_To_Expiry=1.0,OptionType=Call,ExerciseStyle=European
```

The reason for the inclusion of the `Volatility` attribute will be discussed later.

If we explicitly passed all the information required by the underlying C function `black_scholes` then the form of the required entry (referred to here as E_2) would be:

```
Trade=EquityOptionDeal,Reference=EQ:WebComm-1004A,Equity=WebComm,Volatility=0.1,_
Strike=95.0,Time_To_Expiry=1.5,OptionType=Put,ExerciseStyle=European,EquitySpot=100,_
FXEquityCurrency=0.5565,RiskFreeRate=0.1,DividendYield=0.05
```

The reason that E_1 does not require the extra four deal attributes `EquitySpot=100,FXEquityCurrency=0.5565,RiskFreeRate=0.1,` `DividendYield=0.05` is that these are stored in a *market data dictionaries* object and are accessed by the C# class `EquityOptionDeal` before the C function `black_scholes` is called.

The market data dictionaries are populated as soon as the application starts. Exhibit 8.2 shows an example market data file. This is a plain text file, and provides a common repository for the market parameters that are required by the

```
// Broad-Investments : Contains equity, FX and IR products
//=========================================================
// Example FX deal Foreign currency = GBP Domestic Currency = USD, Strike = 1.5,
Settlement = 4.0 years
// Note: The Strike is the number of units of domestic currency that have been agreed to be
paid for one unit
// of foreign currency.

Trade=FXForwardDeal,Reference=FX-5001,ForeignAmount=100,Strike=1.5,ForeignCurrency=GBP,_
DomesticCurrency=USD,Settlement=4.0,BuySell=Buy

Trade=ForwardRateAgreementDeal,Reference=IR-6001,Principal=100.0,Strike=3.0,Currency=GBP,_
Maturity=4.5,Start=4.0,BuySell=Buy

//Trade=FXOptionDeal,Reference=FXOption_Call,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,_
Time_To_Expiry=5.0,ExerciseStyle=European,OptionType=Call,BuySell=Buy

Trade=DownOutEquityOptionDeal,Reference=Tech-7001,Equity=Real-Computers,Volatility=0.2,_
Strike=100.0,Time_To_Expiry=1.0,Barrier_Level=90.0,OptionType=Call,CalcMethod=Analytic
/*
Trade=DownOutEquityOptionDeal,Reference=Tech-7002,Equity=Real-Computers,Volatility=0.2,_
Strike=100.0,Time_To_Expiry=1.0,Barrier_Level=90.0,CalcMethod=MonteCarlo,OptionType=Call,_
NumberScenarios=10000,UseBrownianBridge=true

Trade=DownOutEquityOptionDeal,Reference=Tech-7005,Equity=Real-Computers,Volatility=0.2,_
Strike=100.0,Time_To_Expiry=1.0,Barrier_Level=90.0,CalcMethod=Analytic,OptionType=Call
*/
Trade=DownOutEquityOptionDeal,Reference=Tech-7006,Equity=Real-Computers,Volatility=0.2,_
Strike=100.0,Time_To_Expiry=1.0,Barrier_Level=90.0,CalcMethod=Numeric,OptionType=Call

Trade=DownOutEquityOptionDeal,Reference=Tech-7007,Equity=Real-Computers,Volatility=0.2,_
Strike=100.0,Time_To_Expiry=1.0,Barrier_Level=90.0,CalcMethod=Numeric,OptionType=Call,_
ExerciseStyle=American

//Trade=DownOutEquityOptionDeal,Reference=JPCA_BB_False,Equity=Real-Computers,Strike=100.0,_
Time_To_Expiry=1.0,Barrier_Level=90.0,CalcMethod=MonteCarlo,OptionType=Call,_
NumberScenarios=10000,UseBrownianBridge=false

Trade=DownOutFXOptionDeal,Reference=FX-5004,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,Time_To_Expiry=5.0,ExerciseStyle=European,_
OptionType=Call,BuySell=Buy,CalcMethod=Analytic,Barrier_Level=0.01

Trade=DownOutFXOptionDeal,Reference=FX-5006,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,Time_To_Expiry=5.0,ExerciseStyle=European,_
OptionType=Call,BuySell=Buy,CalcMethod=MonteCarlo,UseBrownianBridge=true,Barrier_Level=0.01

// American FX Barrier Call
Trade=DownOutFXOptionDeal,Reference=FX-5007,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,Time_To_Expiry=5.0,OptionType=Call,BuySell=Buy,_
CalcMethod=Numeric,Barrier_Level=0.01,ExerciseStyle=American

// European Put
Trade=FXOptionDeal,Reference=FX-5008,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,Time_To_Expiry=5.0,ExerciseStyle=European,_
OptionType=Put,BuySell=Buy

Trade=DownOutFXOptionDeal,Reference=FX-5009,NumberOfUnits=123,Strike=0.5,Volatility=0.1375,_
ForeignCurrency=USD,DomesticCurrency=GBP,Time_To_Expiry=5.0,OptionType=Put,BuySell=Buy,_
CalcMethod=MonteCarlo,Barrier_Level=0.01,ExerciseStyle=European
```

Exhibit 8.1 Here we show an example portfolio driver file and the individual portfolio definition files `EQ-Investments.txt` and `Broad-Investments.txt`. The symbol _ is used to indicate a line continuation; it should be noted the C# application requires each deal to be specified on a *single* line.

deal classes. For instance, all foreign exchange derivatives will need access to the current FX rates, and all equity derivatives will require the current equity price. The contents of the market data file can be updated as frequently as required

```
// Currency market data. This is used for pricing interest rate swaps, FX options, etc...

Currency.USD,FXSpot=0.5565,YieldCurve=[(0.0027,0.0184),...,(40.58,0.0533)],
VolCurve=[(0.0,0.10),(1.0,0.12),(3.0,0.13),(6.0,0.14),(20,0.15)]
Currency.GBP<<--Base,FXSpot=1.0,YieldCurve=[(0.0027,0.047),...,(50.03,0.042)],
VolCurve=[(0.0,0.10),(1.0,0.12),(3.0,0.13),(6.0,0.14),(20,0.15)]
Currency.EUR,FXSpot=0.689024,YieldCurve=[(0.0,0.04),...,(20,0.056)],
VolCurve=[(0.0,0.10),(1.0,0.12),(3.0,0.13),(6.0,0.14),(20,0.15)]
Currency.CAD,FXSpot=1.5,YieldCurve=[(0.0,0.04),...,(20,0.056)],
VolCurve=[(0.0,0.10),(1.0,0.12),(3.0,0.13),(6.0,0.14),(20,0.15)]

// Equity market data. This is used for pricing equity options, etc...

Equity.Imperial-Art,Currency=GBP,Spot=9.0,DivYield=0.03
Equity.Real-Computers,Currency=USD,Spot=200.0,DivYield=0.04
Equity.TelComm,Currency=GBP,Spot=120.0,DivYield=0.09
Equity.WebComm,Currency=USD,Spot=100.0,DivYield=0.07
Equity.Hackers,Currency=GBP,Spot=40.0,DivYield=0.02
Equity.LaserComm,Currency=GBP,Spot=95.0,DivYield=0.05
Equity.SmartWeb,Currency=GBP,Spot=100.0,DivYield=0.01
Equity.Web-Comm,Currency=GBP,Spot=100.0,DivYield=0.04
Equity.Mobile-Tech,Currency=GBP,Spot=92.0,DivYield=0.02
Equity.CompuKalc,Currency=GBP,Spot=95.0,DivYield=0.11
Equity.The-Bookshop,Currency=GBP,Spot=100.0,DivYield=0.02
Equity.Everyman-Books,Currency=GBP,Spot=100.0,DivYield=0.03
Equity.The-RealBook-Company,Currency=GBP,Spot=100.0,DivYield=0.04
Equity.Drinks-4U,Currency=GBP,Spot=100.0,DivYield=0.05
Equity.Beverage-Ltd,Currency=GBP,Spot=100.0,DivYield=0.06
Equity.H2O-Ltd,Currency=GBP,Spot=100.0,DivYield=0.05
Equity.Fine-Wines-Ltd,Currency=GBP,Spot=100.0,DivYield=0.03
Equity.French-Wines-Ltd,Currency=EUR,Spot=100.0,DivYield=0.2
Equity.The-English-Beer-Company,Currency=GBP,Spot=100.0,DivYield=0.03
Equity.Water-Works-Ltd,Currency=GBP,Spot=100.0,DivYield=0.012
Equity.Welsh-Spring,Currency=GBP,Spot=100.0,DivYield=0.06
Equity.ThamesBeer,Currency=GBP,Spot=100.0,DivYield=0.05
Equity.Edingburgh-Whiskey,Currency=GBP,Spot=100.0,DivYield=0.04
Equity.The-Wine-Box,Currency=GBP,Spot=100.0,DivYield=0.085

// Bond market data. This is used for pricing bonds, and credit derivatives such as CDS, and
TRS

Bond.Fine-Wines-Ltd-Bond-2020,Currency=GBP,Spot=150.0,
SurvivalProb=[(0.0,1.0),(1.0,0.9),(3.0,0.96),(6.0,0.9),(20,0.5)]
Bond.Hackers-Bond-2018,Currency=GBP,Spot=200.0,
SurvivalProb=[(0.0,1.0),(2.0,0.91),(5.0,0.9),(8.0,0.8),(30,0.6)]
Bond.Hackers-Bond-2060,Currency=GBP,Spot=260.0,
SurvivalProb=[(0.0,1.0),(1.0,0.92),(20.0,0.8),(20.0,0.65),(60,0.7)]
Bond.Real-Computers-Bond-2020,Currency=USD,Spot=100.0,
SurvivalProb=[(0.0,1.0),(1.0,0.94),(4.0,0.9),(8.0,0.6),(30,0.5)]

// Market data correlation. These are used for multiasset options

Correlation.Imperial-Art,Real-Computers=0.5

Correlation.Real-Computers,WebComm=0.4
Correlation.Real-Computers,Hackers=0.5
Correlation.Real-Computers,LaserComm=0.3
Correlation.Real-Computers,SmartWeb=0.4

Correlation.TelComm,Hackers=0.5
Correlation.TelComm,LaserComm=0.5
Correlation.TelComm,SmartWeb=0.5
Correlation.TelComm,Web-Comm=0.5
Correlation.LaserComm,SmartWeb=0.5

Correlation.Hackers,Mobile-Tech=0.4
Correlation.LaserComm,Mobile-Tech=0.4
Correlation.SmartWeb,Mobile-Tech=0.5
Correlation.Web-Comm,Mobile-Tech=0.5
```

Exhibit 8.2 An example market data file, which is used to specify the current market values such as equity spot, FX spot, interest rate yield curves, etc. The third line in the file provides currency information for GBP, and the entry `Currency.GBP«--Base` specifies that the base currency will be GBP, and thus all portfolio and deal values will be computed in GBP.

```
Correlation.Mobile-Tech,The-Bookshop=0.1
Correlation.CompuKalc,LaserComm=0.3

Correlation.ThamesBeer,French-Wines-Ltd=0.3
Correlation.ThamesBeer,Fine-Wines-Ltd=0.5
Correlation.ThamesBeer,H2O-Ltd=0.5
Correlation.ThamesBeer,Beverage-Ltd=0.6
Correlation.ThamesBeer,Drinks-4U=0.6
Correlation.ThamesBeer,The-RealBook-Company=0.8
Correlation.ThamesBeer,People-Books=0.1

                      .    .    .

Correlation.Edingburgh-Whiskey,French-Wines-Ltd=0.5
Correlation.Edingburgh-Whiskey,Water-Works-Ltd=0.5
Correlation.Edingburgh-Whiskey,The-English-Beer-Company=0.6
```

Exhibit 8.2 (*Continued*).

(i.e., daily, hourly, etc.) but will always maintain a set of market values that can be used consistently across all deal valuations.

It should be mentioned that the main advantage of type E_1 deal entries is not just that the portfolio definition file is smaller—it also ensures that consistent market data values are used to price all the trades in the portfolio. When type E_2 deal entries are used, it is necessary to ensure that all the extra deal attributes are updated as new market data becomes available. This would be a time-consuming task and, if only a partial update occurs, could give rise to invalid deal valuations caused by inconsistent deal attribute values such as `FX-EquityCurrency`, `DividendYield`.

The format of the result files is shown in Exhibit 8.3. The output syntax is simply:

```
<deal value>=<deal reference>,<deal type>
```

It can be seen that each deal is valued in base currency (which here is specified in the market data file as GBP) and the total value for the portfolio is also reported.

Results file for portfolio EQ-investments

```
============================
EQ-Investments in units of GBP
EQ-Investments   :14/07/2007 00:00:00
============================
 4.3501=EQ:LaserComm-1001,Equity Option
 2.9278=EQ:LaserComm-1002,Equity Option
 3.5716=EQ:LaserComm-1003,Equity Option
 2.0245=EQ:LaserComm-1004,Equity Option
 2.2171=EQ:WebComm-1004A,Equity Option
 41.7119=EQ:CompuKalc-1005,Rainbow option(two equities)
 14.0274=EQ:MobileTech|CompuKalc-1006,Rainbow option(two equities)
 8.8511=EQ:MobileTech|CompuKalc-1007,Rainbow option(two equities)
 70.3151=EQ:MobileTech|TelComm-1008,Rainbow option(two equities)
 13.3263=JPCAM11,Rainbow option(two equities)
 6.5840=JPCAM111,Rainbow option(two equities)
 72.7866=Tech-10001,Three Equity Option
 70.1046=Tech-10002,Three Equity Option
 69.7412=Tech-10003,Three Equity Option
 1010.6123=Tech-10004,Three Equity Option
 1030.3894=Tech-10005,Three Equity Option
 2850.8918=Tech-10006,Three Equity Option
```

```
 2838.9643=Tech-10007,Three Equity Option
 2838.9643=Tech-10008_Generic,Generic Equity Option
 399.4981=Tech-10009,Three Equity Option
 409.1153=Tech-10010,Three Equity Option
 2646.6473=Drinks-20001,Four Equity Option
 2644.3642=Drinks-20002,Four Equity Option
 15.3381=Drinks-20003,Four Equity Option
 21.8691=Drinks-20004,Four Equity Option
 53.8106=Drinks-20005,Four Equity Option
 63.2889=Drinks-20006,Four Equity Option
 1511.6543=Drinks-20007,Four Equity Option
 1524.5000=Drinks-20008,Four Equity Option
 1510.7045=Drinks-20009,Four Equity Option
 1518.8670=Drinks-20010,Four Equity Option
 1513.7578=Drinks-20011,Four Equity Option
 1524.5000=Drinks-20012,Generic Equity Option
 1513.7578=Drinks-20013,Four Equity Option
 2030.2451=Drinks-20013,Generic Equity Option
 6.1238=JPCA_111,Down Out Equity Option
 6.1240=JPCA_111A,Down Out Equity Option
 3.0006=JPCA_112,Down Out Equity Option
 3.0006=JPCA_113,Down Out Equity Option
 3.0006=JPCA_114,Down Out Equity Option
 3.0036=JPCAPP_115,Double Knock Out Call Equity Option
========================
TOTAL VALUE =  29878.53 GBP
========================
```

Results file for portfolio broad-investments

```
============================
Broad-Investments in units of GBP
Broad-Investments   :14/07/2007 00:00:00
============================
 9.4359=FX-5001,FX Forward
-9.4359=FX-5002,FX Forward
 0.8661=IR-6001,Forward Rate Agreement
 1.1755=IR-6002,Forward Rate Agreement
 52.6353=Tech-7001,Down Out Equity Option
 52.6757=Tech-7002,Down Out Equity Option
 52.6757=Tech-7003,Down Out Equity Option
 52.6583=Tech-7004,Down Out Equity Option
 52.6353=Tech-7005,Down Out Equity Option
 52.6358=Tech-7006,Down Out Equity Option
 55.6500=Tech-7007,Down Out Equity Option
 11.6849=FX-5003,FX Option
 11.6849=FX-5004,Down Out FX Option
 11.6813=FX-5005,Down Out FX Option
 11.5356=FX-5006,Down Out FX Option
 11.9998=FX-5007,Down Out FX Option
 2.9173=FX-5008,FX Option
 2.9827=FX-5009,Down Out FX Option
========================
TOTAL VALUE =  438.09 GBP
========================
```

Exhibit 8.3 Portfolio results files. The reporting currency is set in the market data file; in this example, all values are given in pounds sterling.

8.2 Storing and retrieving the market data

As mentioned before, the market data required to price derivatives is stored in market data dictionaries. The `MarketDataDictionaries` class, shown in

Code excerpt 8.4, contains a set of C# Dictionary member items which hold all the market data required by the deal classes. Below we give the declaration of the `CurrencyTable`, `EquityTable` and `CorrelationTable` dictionaries:

```
public static Dictionary<string, Currency> CurrencyTable = new Dictionary<string,_
Currency>();
public static Dictionary<string, Equity> EquityTable = new Dictionary<string,_
Equity>();
public static Dictionary<string, Correlation> CorrelationTable = new Dictionary<string,_
Correlation>();
```

Each dictionary entry is made up of a {`<unique-key>, <value-object>`} pair, where `unique-key` is a unique string, and `value-object` is a class containing the corresponding market data. We will now consider each of the above dictionaries and the information they hold in more detail.

8.2.1 `CurrencyTable`

In a `CurrencyTable` dictionary `value-object` is a class of type `Currency` and is used to store currency information. The class declaration is provided in Code excerpt 8.3.

```
public class TPair : IComparable
{
    public double t;
    public double val;

    public TPair(double t1, double val1)
    {
        t = t1;
        val = val1;
    }

    int IComparable.CompareTo(object obj)
    {
        TPair temp = (TPair)obj;
        if (this.t > temp.t)
            return (1);
        if (this.t < temp.t)
            return (-1);
        else
            return (0);
    }
}

// ICurve - a curve for storing interest rates
public class ICurve: List<TPair>
{
    private double t_pt;  // internal value used for matching
    private string name_ = "";

    public ICurve (string name) {
        name_ = name;
    }

    public double this[double t_0, double t]
    { get { // return the discount factor between t and t1

        double eps = 1.0e-6;
        double val;
```

Code excerpt 8.3 Code showing the class `Currency` and also the classes `ICurve` and `TPair` which all enable the interest rate yield curve to be stored.

```
            t_pt = t;
            // find the bounding indicies corresponding to a given t value
            int indx1 = this.FindIndex(TPairFind);

            if (indx1 == -1)
            throw new Exception("Invalid market data interest rate yield curve for currency "
                                                                                    + name_);

            double v2 = this[indx1].val;
            double v1 = this[indx1 - 1].val;
            double t2 = this[indx1].t;
            double t1 = this[indx1 - 1].t;

            if (Math.Abs(t2 - t) < eps)
            {
                val = v2;
            }
            else
            { // use linear interpolation to compute the value of DF
                val = v1 + ((v2 - v1) / (t2 - t1)) * (t - t1);
            }

            val = Math.Exp(-val * t);

            return val;
          }
       }

        public double this[double t, double t1, double t2] {
        get {
            // return the forward rate between t1 and t2
            // t2 >= t1
             double DF1 = this[t, t1];
             double DF2 = this[t, t2];
             double fwd = (DF1 / DF2 - 1.0) / (t2 - t1);

             return fwd;
          }
        }

        private bool TPairFind(TPair v)
        {
            if ((v.t >= t_pt))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }

    public class Currency
    {
        public string name;           // the name of the currency (e.g GBP, or USD)
        public double spot;           // the FX spot of the currency with respect to base
                                         currency
        public ICurve YieldCurve;     // the currency yield curve
        public VCurve VolCurve;       // the volatility of the yield curve (not currently used
                                         by the C# application)
        public Currency(string name1, double spot1) // two parameter constructor
        {
            name = name1;
            spot = spot1;
            YieldCurve = new ICurve(name1);
        }

        public Currency(string name1) // single parameter constructor
        {
            YieldCurve = new ICurve(name1);
            VolCurve = new VCurve(name1);
        }
    }
```

Code excerpt 8.3 (*Continued*).

```
string CcyCode = "USD"; // set the unique currency code to USD

// create a new (empty) entry in the CurrencyTable for USD
CurrencyTable.Add(CcyCode, new Currency(CcyCode));

double FXSpot = 0.5565; // set the USD to base currency FX spot

// assign to public data member spot, in class Currency
CurrencyTable[CcyCode].spot = FXSpot;
// assign to public data member name, in class Currency
CurrencyTable[CcyCode].name = CcyCode;

double t;
double rt;

t= 0.0027;      // time - for point 1
rt = 0.0184;   // rate - for point 1
 // add the first point to the public data member YieldCurve, in class Currency
CurrencyTable[CcyCode].YieldCurve.Add(new TPair(t, rt));
        .   .   .
t = 40.58;
rt = 0.0533;
CurrencyTable[CcyCode].YieldCurve.Add(new TPair(t, rt));
```

Code excerpt 8.4 C# code showing the addition of USD currency market data to the `CurrencyTable` dictionary.

It is straightforward to add currency data to `CurrencyTable`. The C# code fragment shown in Excerpt 8.4 illustrates the addition of `USD` information.

The information in the `CurrencyTable` is accessed by the deal classes and used to compute discount factors, forward rates and FX spots. In the market data file (see Exhibit 8.2) the YieldCurve consists of a set of time/value pairs and is defined using the following syntax:

YieldCurve $= [(t_1, r_1), \ldots, (t_i, r_i), \ldots, (t_n, r_n)]$, where t_i is the time in years, and r_i is the corresponding zero coupon rate with tenor t_i.

The value, at $t = 0$, of a zero coupon bond with unit cash flow at maturity, t_i, is $\exp(-r_i\ t_i)$, and is known as the discount factor $DF(0, t_i)$. Code excerpt 8.4 shows the addition of two data items to the USD yield curve, while Code excerpt 8.5 illustrates the retrieval from $DF(0, t_i)$ of r_i. The code fragment:

```
double discount_fac = DF[0, Time_To_Expiry_];
RiskFreeRate = -Math.Log(discount_fac) / Time_To_Expiry_;
```

computes the zero coupon rate `RiskFreeRate` and this can be used as a value for the risk free rate required by the option pricing routines in `Analytics_MathLib.dll`. It should be mentioned that it would have been more efficient to have written code to directly obtain the interpolated risk free rate from the USD yield curve (without first computing the associated discount factor). However, the required code can easily be supplied by the reader.

Discount factors and forward rates are accessed from an `ICurve` object. The discount factor is obtained as follows:

```
ICurve DF = MarketDataDictionaries.
            CurrencyTable[Currency_].YieldCurve;
```

```
try
{
   string Currency_ = "USD";
   // DF  will be used to access discount factors
   ICurve DF =  MarketDataDictionaries.CurrencyTable[Currency_].YieldCurve;

   // FWD will be used to access forward rates
   ICurve FWD =  MarketDataDictionaries.CurrencyTable[BaseCurrency].YieldCurve;

   // obtain the discount factor DF(0,1) using methods in class ICurve
   double discount_fac = DF[0, Time_To_Expiry_];

   RiskFreeRate = -Math.Log(discount_fac) / Time_To_Expiry_;
   double Time_To_Expiry_ = 1.0;
   double FXspot =

  // obtain the FX spot with repect the base currency (GBP)
   MarketDataDictionaries.CurrencyTable[Currency_].spot;


   double t1 = 1.0;;
   double t2 = 1.5;

   // obtain the forward rate F(0,1,1.5) using methods in class ICurve
   double forward_rate = FWD[0, t1, t2];
}
catch
{
   throw new Exception(Pre_string_ + "--- No Market Data supplied for " + Currency_);
}
```

Code excerpt 8.5 Code showing the retrieval of USD currency market data from the CurrencyTable dictionary.

```
public class Equity
{
    public string Name;
    public double Spot;
    public string Currency;
    public double DivYield;

    public Equity(string Name1, double Spot1, string Currency1, double DivYield1)
    {
        Name = Name1;
        Spot = Spot1;
        Currency = Currency1;
        DivYield = DivYield1;
    }

    public Equity()
    {
    }
}

public class Correlation
{
    public string Name1;
    public string Name2;
    public double Correl;

    public Correlation(string Name11, string Name12, double Correl1)
    {
        Name1 = Name11;
        Name2 = Name12;
        Correl = Correl1;
    }
    public Correlation()
    {
    }
}
```

Code excerpt 8.6 The equity and correlation classes.

declares the `ICurve` object `DF`. Then the discount factor between `0` and `Time_To_Expiry_` is computed with the statement:

```
double discount_fac = DF[0, Time_To_Expiry_];
```

where `DF[0, Time_To_Expiry_]` calls the `ICurve` accessor (declared as `public double this[double t_0, double t]`) with `t_0 = 0` and `t = Time_To_Expiry`. It can be seen from Code excerpt 8.3 that linear interpolation is performed by the accessor if required.

The forward rate is obtained in a similar manner:

```
ICurve FWD = MarketDataDictionaries.
             CurrencyTable[Currency_].YieldCurve;
```

declares the `ICurve` object `FWD`. Then the forward rate $F(0, t_1, t_2)$ is returned with the statement:

```
double forward_rate = FWD[0, 0.5, 1.5];
```

where `FWD[0, t1, t2]` calls the `ICurve` accessor (declared as `public double this[double t, double t1, double t2]`) with `t = 0`, `t1=0.5`, and `t2 = 1.5`. It can be seen from Code excerpt 8.3 that the accessor computes the forward rate as:

```
FWD[0,t1,t2] = (DF[0,t1] / DF[0,t2] - 1.0)
               / (t2 - t1);
```

`EquityTable` and `CorrelationTable`

Code excerpt 8.7 shows how market data is added to the internal dictionaries of the `MarketDataDictionaries` class. For instance, to add a correlation

```
public class MarketDataDictionaries
{
    public static Dictionary<string, Currency> CurrencyTable = new Dictionary<string,
    Currency>();
    public static Dictionary<string, Equity> EquityTable = new Dictionary<string,
    Equity>();
    public static Dictionary<string, Correlation> CorrelationTable = new Dictionary<string,
    Correlation>();
    protected static string BaseCurrency_ = "";

    public static string GetBaseCurrency()
    {
        return BaseCurrency_;
    }

    public static void LoadMarketData(string marketdata_file)
    {
        // Load the market data file and assign values to dictionaries
        StreamReader MDFILE = new StreamReader(marketdata_file);
        string cur_line = "";

        while ((cur_line = MDFILE.ReadLine()) != null)  // loop through the market data
                                                           file
```

Code excerpt 8.7 The `MarketDataDictionaries` class, illustrating how market data is added to the internal dictionaries.

```
{
    if ((cur_line.Length > 1) && ((cur_line.Substring(0, 2) != "//")))
    {
        char[] seps = new char[] { '=', ',' };
        string[] v = cur_line.Split(seps, StringSplitOptions.None);

        int num_elems = v.GetUpperBound(0);
        int k = 0;
        double t_, val_;
        bool stop;
        double FXSpot = 0.0;

        if (v[0].Substring(0, 8) == "Currency") // currency data
        {
            string CcyCode = v[0].Substring(9, 3);

            if (v[0].IndexOf("<<--Base") != -1) BaseCurrency_ = CcyCode;

            k += 2;

            if (!CurrencyTable.ContainsKey(CcyCode))
            {
                CurrencyTable.Add(CcyCode, new Currency(CcyCode));
            }
            else
            {
                throw new Exception("Spot & interest rate market data already
                                                        supplied for " + CcyCode);
            }

            FXSpot = (double)Convert.ChangeType(v[k], typeof(double));
            CurrencyTable[CcyCode].spot = FXSpot;
            CurrencyTable[CcyCode].name = CcyCode;

            k += 2;
            t_ = (double)Convert.ChangeType(v[k].Substring(2, v[k].Length - 2),
                 typeof(double));
            k += 1;
            val_ = (double)Convert.ChangeType(v[k].Substring(0, v[k].Length - 1),
                   typeof(double));
            CurrencyTable[CcyCode].YieldCurve.Add(new TPair(t_, val_));
            k += 1;
            int vv = v[k].IndexOf(")]");
            stop = false;
            while (!stop) // Add the yield curve data
            {
                t_ = (double)Convert.ChangeType(v[k].Substring(1, v[k].Length - 1),
                     typeof(double));
                k += 1;
                if (v[k].IndexOf(")]") == -1)
                {
                    val_ = (double)Convert.ChangeType(v[k].Substring(0, v[k].
                           Length - 1),typeof(double));
                }
                else
                {
                    val_ = (double)Convert.ChangeType(v[k].Substring(0, v[k].
                    Length - 2), typeof(double));
                    stop = true;
                }
                k += 1;
                CurrencyTable[CcyCode].YieldCurve.Add(new TPair(t_, val_));
            }
            CurrencyTable[CcyCode].YieldCurve.Sort();

                    .      .     .

        }
        else if (v[0].Substring(0, 6) == "Equity") // equity data
        {
            int idx = v[k].IndexOf(".");
            string EquityName = v[0].Substring(idx + 1, v[0].Length - idx - 1);
            k += 2;
            string CcyCode = v[k];
```

Code excerpt 8.7 (*Continued*).

```
                    k += +2;
                    double spot = (double)Convert.ChangeType(v[k], typeof(double));
                    k += +2;
                    double div = (double)Convert.ChangeType(v[k], typeof(double));

                    if (!EquityTable.ContainsKey(EquityName))
                    {
                        EquityTable.Add(EquityName, new Equity());
                        EquityTable[EquityName].Currency = CcyCode;
                        EquityTable[EquityName].Name = EquityName;
                        EquityTable[EquityName].Spot = spot;
                        EquityTable[EquityName].DivYield = div;
                    }
                    else
                    {
                        throw new Exception("Spot & currency market data already supplied
                                                                    for " + EquityName);
                    }
                }
                                        .     .      .

                else if (v[0].Length >= 12 && v[0].Substring(0, 11) == "Correlation")
                // correlation data
                {
                    int idx = v[0].IndexOf(".");
                    string AssetName1 = v[0].Substring(idx + 1, v[0].Length - idx - 1);
                    k += 1;
                    string AssetName2 = v[k];
                    k += 1;
                    double corr = (double)Convert.ChangeType(v[k], typeof(double));
                    string CorrelationKey = AssetName1 + "%" + AssetName2;

                    if(!CorrelationTable.ContainsKey(CorrelationKey))
                    {
                       // ie The-Wine-Box%Water-Works-Ltd and The-Wine-Box%Water-Works-Ltd

                        CorrelationTable.Add(CorrelationKey, new Correlation());
                        CorrelationTable[CorrelationKey].Correl = corr;
                        CorrelationTable[CorrelationKey].Name1 = AssetName1;
                        CorrelationTable[CorrelationKey].Name2 = AssetName2;

                        CorrelationKey = AssetName2 + "%" + AssetName1;
                        CorrelationTable.Add(CorrelationKey, new Correlation());
                        CorrelationTable[CorrelationKey].Correl = corr;
                        CorrelationTable[CorrelationKey].Name1 = AssetName2;
                        CorrelationTable[CorrelationKey].Name2 = AssetName1;
                    }
                    else
                    {
                        throw new Exception("market data already supplied for "
                                                          + CorrelationKey);
                    }
                }
            }
        }
        MDFILE.Close();
    }
}
```

Code excerpt 8.7 (*Continued*).

entry it is first necessary to construct the dictionary key, and then determine whether or not the entry already exists in the dictionary. This is shown in the code fragment below:

```
// first construct the unique key string from AssetName1 and AssetName2

  string CorrelationKey = AssetName1 + "%" + AssetName2;

  // Now check whether this key already exists in the dictionary CorrelationTable.
  // If it doesn't then add a new entry, if it does then raise an exception

  if(!CorrelationTable.ContainsKey(CorrelationKey))
```

```
{
      // Create a new empty dictionary entry for the unique key
      CorrelationTable.Add(CorrelationKey, new Correlation());

      // Now fill out the entry by assigning the correlation and the asset names
      CorrelationTable[CorrelationKey].Correl = corr;
      CorrelationTable[CorrelationKey].Name1 = AssetName1;
      CorrelationTable[CorrelationKey].Name2 = AssetName2;

      // Create another empty dictionary entry with the asset names in reverse order
      // (because correlation(a,b) = correlation(b,a)

      CorrelationKey = AssetName2 + "%" + AssetName1;

      // Now fill out the entry by assigning the correlation and the asset names
       CorrelationTable.Add(CorrelationKey, new Correlation());
       CorrelationTable[CorrelationKey].Correl = corr;
       CorrelationTable[CorrelationKey].Name1 = AssetName2;
       CorrelationTable[CorrelationKey].Name2 = AssetName1;
}
else
{
   throw new Exception("market data already supplied for " + CorrelationKey);
}
```

8.3 The `PricingUtils` class and the `Analytics_MathLib`

It has already been mentioned that the deal classes often make calls to the C pricing function that are contained in `Analytics_MathLib.dll`. To do this it is necessary to provide a C# declaration for the corresponding C function. This is carried out in the `PricingUtils` class by declaring the routines as `extern` and using the C# `DllImport` attribute, which is defined in the namespace `System.Runtime.InteropServices`. Code excerpt 8.8 provides a sample of the C# declarations in `PricingUtils` and some of the corresponding C dec-

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.IO;
using System.Text;
using System.Runtime.InteropServices;

   public class PricingUtils
   {
      public static double EPS = 1.0e-6;

      [DllImport("Analytics_MathLib.dll")]
      public static extern void dko_call(double lower_barrier, double upper_barrier,
             double theta_m, double S0, ref double sigma_array, ref double sigma_times,
             int n_sigma, double r, double opt_mat, double X, int is_american, ref double
             option_value,
             IntPtr greeks, double q, int ns_below_S0, int ns_above_S0, int nt, ref int iflag);

      [DllImport("Analytics_MathLib.dll", EntryPoint = "black_scholes")]
      public static extern void black_scholes2(ref double value, ref double greeks, double s0,
                                    double x, double sigma,
                                    double t, double r, double q, int put, ref int iflag);
```

Code excerpt 8.8 The PricingUtils class, which permits C# code to call the C functions that reside in `Analytics_Mathlib.dll`. The attribute `DllImport` (defined in namespace `System.Runtime.InteropServices`) indicates to C# Interop services that unmanaged code is being called.

```
    [DllImport("Analytics_MathLib.dll", EntryPoint = "black_scholes")]
    public static extern void black_scholes(ref double value, IntPtr greeks, double s0,
                            double x, double sigma,
                            double t, double r, double q, int put, ref int iflag);

    [DllImport("Analytics_MathLib.dll")]
    public static extern void opt_gfd(double theta, double S0, double sigma, double r,
                          double T, double X, int is_american,
                          int put, ref double value, IntPtr greeks, double q, int ns, int nt,
                          double smax, ref int iflag);

    [DllImport("Analytics_MathLib.dll")]
    public static extern void standard_2D_binomial(ref double value, double S1, double S2,
              double X, double sigma1,
              double sigma2, double rho, double T, double r, double q1, double q2, int put,
              int M, int is_max,
              int is_american, ref int iflag);

                    .        .       .

    public static double RndNorm(double mean, double std)
    {
        return mean + std * normal(0.0, 1.0);
    }

    [DllImport("Analytics_MathLib.dll")]
    public static extern double normal(double mean, double std);

    [DllImport("Analytics_MathLib.dll")]
    public static extern void set_seed(int seed);

    [DllImport("Analytics_MathLib.dll", EntryPoint = "multivariate_normal2")]
    public static extern void multivariate_normal(int is_fcall, ref double a, int n,
                                      ref double c, int tdc, double eps,
                                      ref double r, ref double z, ref int iflag);
    // array r has size nr = ((n+1)*(n+2))/2 + 1
}
```

Code excerpt 8.8 (*Continued*).

larations are given in Code excerpt 8.9. It can be seen that C function parameters passed by value and declared as `double dval` and `long ival` correspond to C# `double dval` and `int ival` parameters, respectively. C functions that are passed by reference are a little more tricky to deal with in C#. This is because C# does not *explicitly* support pointers. A C `double` parameter passed by reference using `double *dval` can be mapped in C# by using `ref double dval`. Similarly, the C function parameter `long* ival` is declared in C# as `ref int ival`. It should be noted that for clarity we use the C syntax `double a_dval[]` and `long a_ival[]` for arrays of `double` and `long`, respectively; it is also correct to declare these as `double *a_dval`, and `long *a_ival`.

However, there is still a problem because C functions can be coded to check whether or not a `null` pointer has been supplied as a parameter, and then take appropriate action. For instance, the function `black_scholes` in Code excerpt 4.1, will not compute the Greeks if the parameter `double greeks[]` is `null`. The code fragment below shows how `IntPtr`, which is a recent addition to C#, can be used to resolve this:

```
[DllImport("Analytics_MathLib.dll", EntryPoint = "black_scholes")]
public static extern void black_scholes2(ref double value, ref double greeks,_
double s0, double x,double sigma, double t, double r, double q, int put, ref int iflag);
```

```
[DllImport("Analytics_MathLib.dll", EntryPoint = "black_scholes")]
public static extern void black_scholes(ref double value, IntPtr greeks, double s0,_
double x, double sigma, double t, double r, double q, int put, ref int iflag);
```

The first declaration specifies that the function called `black_scholes` in `Analytics_MathLib.dll` will have the name `black_scholes2` in C#. In addition the C parameter `double greeks[]` is mapped to `ref double greeks` in C#. This means that we will need to declare an array variable (say `the_greeks` which will hold the five Greeks and pass it to `black_scholes2` using the syntax `ref the_greeks[0]`. In this case the Greeks will always be computed, even if we don't want to use them.

The second declaration specifies that the function `black_scholes2` in C# will also call the routine `black_scholes` in `Analytics_MathLib.dll`. However, in this case, the C parameter `double greeks[]` is mapped to `IntPtr greeks` in C#. This allows us to set the parameter `double greeks[]` to `null` by passing the value `IntPtr.Zero`—see for example class `EquityOptionDeal` in Code excerpt 8.10.

```
#define DLLExport __declspec(dllexport)

void DLLExport __stdcall black_scholes(double *value, double greeks[],double s0,double x,
                 double sigma,
                 double t,double r,double q, long put, long *iflag)
{
                              .     .    .

                    See code excerpt 4.1 for more detail

                              .     .    .

}

void DLLExport __stdcall opt_gfd(double theta_m, double asset_price, double sigma, double r,
                 double T,
                 double strike, long is_american, long put, double *option_value,
                 double greeks[], double q, long pns, long nt, double smax, long *iflag)
{
                              .     .    .

                    See code excerpt 5.18 for more detail

                              .     .    .
}

void DLLExport __stdcall multivariate_normal2(long is_fcall, double a[], long n, double c[],
                                              long tdc,
                                              double eps, double r[], double z[], long *iflag)
{
                              .     .    .

                     Standard C code to generate a multivariate normal

                              .     .    .
}
```

Code excerpt 8.9 Illustrative C code which is contained in the windows dynamic link library `Analytics_MathLib.dll`.

```
using System;

namespace Computational_Lib
{
    public class EquityOptionDeal: BaseDeal
    {
        public string Equity { get { return EquityName_; } set { EquityName_ = value; } }
        public int NumberOfUnits { get { return NumberOfUnits_; }_
           set { NumberOfUnits_ = value; } }
        public double Time_To_Expiry { get { return Time_To_Expiry_; }_
           set { Time_To_Expiry_ = value; } }
        public PutCall OptionType{ get {return OptionType_;} set { OptionType_= value;}}
        public EuropeanAmerican ExerciseStyle { get { return ExerciseStyle_; }_
           set { ExerciseStyle_ = value; } }
        public double Strike { get { return Strike_; } set { Strike_ = value; } }
        public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }
        public double Volatility { get { return Sigma_; } set { Sigma_ = value; } }

        protected PutCall  OptionType_ = PutCall.Put;
        protected EuropeanAmerican ExerciseStyle_ = EuropeanAmerican.European;
        protected double   Strike_ = 0;
        protected string   EquityName_ = "";
        protected double   Time_To_Expiry_ = 0.0;
        protected string   Currency_ = "";
        protected double   DividendYield_ = 0.0;
        protected string   Pre_string_ = "";
        protected BuySell  BuySell_ = BuySell.Buy;
        protected int NumberOfUnits_ = 1;
        protected double Sigma_ = 0.0;

        public override string Name()
        {
            return "Equity Option";
        }

        public override double Price()
        {
           Validate();

           double val=0.0;
           double[] greeks = new double[6];
           double s0 = 0.0;
           double fx_spot = 0.0;
           try
           {
               s0 = MarketDataDictionaries.EquityTable[EquityName_].Spot;
               // get current equity price
               Currency_ = MarketDataDictionaries.EquityTable[EquityName_].Currency;
               // get equity volatility (assumed constant)
               DividendYield_ = MarketDataDictionaries.EquityTable[EquityName_].DivYield;
               // get equity dividend yield
           }

           catch
           {
               throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                                    + EquityName_);
           }
           // get the risk free rate to use
           double discount_fac = 0.0;
           double RiskFreeRate = 0.0;

           try
           {
               ICurve DF = MarketDataDictionaries.CurrencyTable[Currency_].YieldCurve;
               // obtain the discount factor
               discount_fac = DF[0, Time_To_Expiry_];
               RiskFreeRate = -Math.Log(discount_fac) / Time_To_Expiry_;
               fx_spot = MarketDataDictionaries.CurrencyTable[Currency_].spot;
           }
```

Code excerpt 8.10 The complete C# code for class `EquityOptionDeal`, which computes the value of a single equity option.

```
            catch
            {
              throw new Exception(Pre_string_ + "--- No Market Data supplied for " + Currency_);
            }

            int iflag, put;

            iflag = 0;
            put = OptionType_ == PutCall.Put ? 1 : 0;

            if (ExerciseStyle_ == EuropeanAmerican.European)  // use BlackScholes
            {
                // call C routine.
                // Note: A null pointer is supplied so that the Greeks are not computed
                PricingUtils.black_scholes(ref val, IntPtr.Zero, s0, Strike_, Sigma_,
                              Time_To_Expiry_,
                              RiskFreeRate, DividendYield_, put, ref iflag);
            }
            else
            {
               // Use Crank Nicolson
               double theta = 0.5;
               int is_american = 1;
               // fix the geometry of the grid (these values should give "reasonable" results)
               int ns = 50;  // 50 divisions on asset axis
               int nt = 50;  // 50 divisions on time axis
               double smax = 10.0 * s0;

               PricingUtils.opt_gfd(theta, s0, Sigma_,RiskFreeRate, Time_To_Expiry_, Strike_,
                  is_american, put, ref val, IntPtr.Zero, DividendYield_, ns,
                  nt, smax, ref iflag);
            }

            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the_
                   pricing library");

            val *= fx_spot * NumberOfUnits_; // return value in base currency

            return val;
        }

        protected override void Validate()
        {
            Pre_string_ = Name() + "  (" + Reference_ +")";
            if (Time_To_Expiry_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than_
                   zero years");
            }
            if (Strike_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
            }
            if (NumberOfUnits_ < 0)
            {
                throw new Exception(Pre_string_ + "--- The number of units cannot be less_
                   than zero");
            }

            if (Sigma_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- Volatility cannot be less than zero");
            }
        }
    }
}
```

Code excerpt 8.10 (*Continued*).

8.4 Equity deal classes

In this section we provide the C# code for both single and multiasset equity options. The trade attributes correspond to the public properties of the deal class, and attribute default values can be readily found by reading the C# code.

The enumerations used by the deal attributes are declared below.

```
public enum BuySell { Buy, Sell };
public enum PutCall { Put, Call };
public enum EuropeanAmerican { European, American };
public enum MinimumMaximum { Minimum, Maximum };
public enum UseMonteCarlo { Yes, No };
public enum CalculationMethod { Analytic, Numeric, MonteCarlo };
```

and it can be seen that the enumerators have *obvious* names.

It has already been noted that the volatility used by the application to price options is supplied as a deal attribute, rather than being stored as market data. The reason for this is simplicity. In Chapter 4 we mentioned that a volatility surface is required to represent the implied volatility used to price options. Storing the implied volatility would thus require a set of volatility surfaces in the market data file and also multidimensional interpolation to retrieve the volatility applicable to a given option. It was thus decided to supply the volatility as a trade attribute—and update its value appropriately.

8.4.1 Single equity option

Code excerpt 8.10 gives the C# code for class `EquityOptionDeal`, which computes the values of a single equity option.

8.4.2 Option on the price of two equities

Code excerpt 8.11 gives the C# code to compute the value of options that depend on the price of two equities.

```
using System;

namespace Computational_Lib
{
    public class TwoEquityOptionDeal : BaseDeal
    {
        public string Equity1 { get { return EquityName1_; } set { EquityName1_ = value; } }
        public string Equity2 { get { return EquityName2_; } set { EquityName2_ = value; } }
        public double Time_To_Expiry { get { return Time_To_Expiry_; }_
           set { Time_To_Expiry_ = value; } }
        public int NumberOfUnits { get { return NumberOfUnits_; }_
           set { NumberOfUnits_ = value; } }
        public PutCall OptionType { get { return OptionType_; } set { OptionType_ = value; } }
        public MinimumMaximum  MinMax{ get { return MinMax_; } set { MinMax_ = value; } }
        public EuropeanAmerican ExerciseStyle { get { return ExerciseStyle_; }_
           set { ExerciseStyle_ = value; } }
```

Code excerpt 8.11 C# code to compute the value of options which depend on the price of two equities. For example, it is possible to specify whether the option is European or American, and if it is on the minimum or maximum of the equity prices.

```
public double Strike { get { return Strike_; } set { Strike_ = value; } }
public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }
public double Volatility1 { get { return Sigma1_; } set { Sigma1_ = value; } }
public double Volatility2 { get { return Sigma2_; } set { Sigma2_ = value; } }

protected PutCall OptionType_ = PutCall.Put;
protected MinimumMaximum MinMax_ = MinimumMaximum.Maximum;
protected EuropeanAmerican ExerciseStyle_ = EuropeanAmerican.European;
protected double Strike_ = 0;
protected string EquityName1_ = "";
protected string EquityName2_ = "";
protected string Currency_ = "";
protected int NumberOfUnits_ = 1;
protected double Time_To_Expiry_ = 0.0;
protected string Pre_string_ = "";
protected double S1_ = 0.0;
protected double S2_ = 0.0;
protected double Sigma1_ = 0.0;
protected double Sigma2_ = 0.0;
protected BuySell BuySell_ = BuySell.Buy;

public override string Name()
{
    return "Rainbow option(two equities)";
}

public override double Price()
{
    Validate();

    double val = 0.0;
    double rho = 0.0; // default correlation set to zero
    double RiskFreeRate = 0.0;
    double fx_spot = 0.0;

    try
    {
        S1_ = MarketDataDictionaries.EquityTable[EquityName1_].Spot;
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                              + EquityName1_);
    }

    try
    {
        S2_ = MarketDataDictionaries.EquityTable[EquityName2_].Spot;
        S1_ = MarketDataDictionaries.EquityTable[EquityName1_].Spot;
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                             + EquityName2_);
    }

    if(string.Compare(MarketDataDictionaries.EquityTable[EquityName1_].Currency,
              MarketDataDictionaries.EquityTable[EquityName2_].Currency) != 0){

        throw new Exception(Pre_string_ + "--- Currencies for both equities are not
                                                                        the same");
    }

    Currency_ = MarketDataDictionaries.EquityTable[EquityName1_].Currency;

    try
    {
        ICurve DF = MarketDataDictionaries.CurrencyTable[Currency_].YieldCurve;
        // obtain the discount factor
        double discount_fac = DF[0, Time_To_Expiry_];
        RiskFreeRate = -Math.Log(discount_fac) / Time_To_Expiry_;
        fx_spot = MarketDataDictionaries.CurrencyTable[Currency_].spot;
    }
```

Code excerpt 8.11 (*Continued*).

```
            catch
            {
                throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                                    + Currency_);
            }

            string corr_key = EquityName1_ + "%" + EquityName2_;

            if (MarketDataDictionaries.CorrelationTable.ContainsKey(corr_key))
                rho = MarketDataDictionaries.CorrelationTable[corr_key].Correl;
            else
                rho = 0.0;

            int iflag, put, is_max;

            iflag = 0;
            put = OptionType_ == PutCall.Put ? 1 : 0;
            is_max = MinMax_ == MinimumMaximum.Maximum ? 1 : 0;

            if (ExerciseStyle_ == EuropeanAmerican.European)  // use analytic method
            {
               PricingUtils.opt_rainbow_bs_2d(ref val, S1_, S2_, Strike_, Sigma1_, Sigma2_,
                                rho, Time_To_Expiry_, RiskFreeRate, is_max,
                                put, ref iflag);
            }
            else { // use numeric method

              double q1 = 0.0;
              double q2 = 0.0;
              int num_steps = 200; // need to use an even number of time steps for the lattice
              int is_american = 1;
              PricingUtils.standard_2D_binomial(ref val, S1_, S2_, Strike_, Sigma1_, Sigma2_,
                                rho, Time_To_Expiry_, RiskFreeRate,
                                q1, q2, put, num_steps, is_max, is_american, ref iflag);
            }

            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                    pricing library");
            val *= fx_spot * NumberOfUnits_; // return value in base currency
            return val;
        }

        protected override void Validate()
        {
            Pre_string_ = Name() + "  (" + Reference_ + ")";

            if (Time_To_Expiry_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than zero
                                                                        years");
            }
            if (NumberOfUnits_ < 0)
            {
                throw new Exception(Pre_string_ + "--- Number of units cannot be less than
                                                                    zero");
            }
            if (Strike_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
            }
            if (Sigma1_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- Volatility1 cannot be less than zero");
            }
            if (Sigma2_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- Volatility2 cannot be less than zero");
            }
        }
    }
}
```

Code excerpt 8.11 (*Continued*).

8.4.3 Generic equity basket option

Here (see Code excerpt 8.12) we consider the abstract deal class `GenericEquityBasketOptionDeal` which enables its derived classes to value an op-

```
using System;

namespace Computational_Lib
{
    public abstract class GenericEquityBasketOptionDeal : BaseDeal
    {
        public string Equities { set { Equities_ = value; } }
        public string Volatilities { set { Volatilities_ = value; } }
        public double Time_To_Expiry { get { return Time_To_Expiry_; }
                                       set { Time_To_Expiry_ = value; } }
        public int NumberScenarios { get { return NumberScenarios_; }
                                       set { NumberScenarios_ = value; } }
        public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }
        public int NumberOfUnits { get { return NumberOfUnits_; }
                                       set { NumberOfUnits_ = value; } }
        public string Currency { get { return Currency_; } set { Currency_ = value; } }
        public double Strike { get { return Strike_; } set { Strike_ = value; } }

        protected string Equities_ = "";
        protected string Volatilities_ = "";
        protected double Time_To_Expiry_ = 0.0;
        protected double RiskFreeRate_ = 0.0;
        protected string Pre_string_ = "";
        protected double[] S_;

        protected double[] Sigma_;
        protected int NumberScenarios_ = 3000;
        protected double[,] Correlations_;
        protected int n_ = 0;
        protected int NumberOfUnits_ = 1;
        protected string Currency_ = "";
        protected BuySell BuySell_ = BuySell.Buy;
        protected double[] ST_;
        protected double Strike_ = 0.0;
        public override string Name()
        {
            return "Generic Equity Option";
        }

        public abstract double Payoff();

        public override double Price()
        {
            Validate();
            double val = 0.0;
            double fx_spot = 0.0;

            char[] seps = new char[] { '%' };
            string[] EquityNames = Equities_.Split(seps, StringSplitOptions.None);

            n_ = EquityNames.Length;

            ST_ = new double[n_];

            S_ = new double[n_];
            Sigma_ = new double[n_];
            for (int k = 0; k < n_; ++k)
            {
                try
                {
                    S_[k] = MarketDataDictionaries.EquityTable[EquityNames[k]].Spot;
                }
```

Code excerpt 8.12 C# code for the abstract class `GenericEquityBasketOptionDeal`. It contains the abstract method `Payoff()`.

```
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                        + EquityNames[k]);
    }
}

for (int k = 1; k < n_; ++k)
{
    if (string.Compare(MarketDataDictionaries.EquityTable[EquityNames[k - 1]].
    Currency, MarketDataDictionaries.EquityTable[EquityNames[k]].Currency) != 0)
            throw new Exception(Pre_string_ + "--- Not all the currencies are
                                                                the same");
}

Currency_ = MarketDataDictionaries.EquityTable[EquityNames[1]].Currency;

try
{
    ICurve DF = MarketDataDictionaries.CurrencyTable[Currency_].YieldCurve;
    // obtain the discount factor
    double discount_fac = DF[0, Time_To_Expiry_];
    RiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
    fx_spot = MarketDataDictionaries.CurrencyTable[Currency_].spot;
}
catch
{
    throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                          + Currency_);
}

string[] Vols = Volatilities_.Split(seps, StringSplitOptions.None);
int n_v;
n_v = Vols.Length;

if (n_v != n_)
    throw new Exception(Pre_string_ + "--- Number of volatilities is not the same
                                                  as the number of equities ");

Sigma_ = new double[n_];
for (int k = 0; k < n_; ++k)
{
    try
    {
        Sigma_[k] = double.Parse(Vols[k]);
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- Invalid volatility supplied for  "
                                                              + EquityNames[k]);
    }
}

Correlations_ = new double[n_, n_];

for (int i = 0; i < n_; ++i)
{
    for (int j = 0; j < n_; ++j)
    {
        if (i != j)
        {
          string corr_key = EquityNames[i] + "%" + EquityNames[j];
          if (MarketDataDictionaries.CorrelationTable.ContainsKey(corr_key))
              Correlations_[i, j] = MarketDataDictionaries
                                    .CorrelationTable[corr_key].Correl;
          else
             Correlations_[i, j] = 0.0; // default correlation is zero
        }
        else
        {
            Correlations_[i, j] = 1.0;
        }
    }
}
```

Code excerpt 8.12 (*Continued*).

```
    int iflag=0;

    val = MonteCarloSim(ref iflag);

    if (iflag != 0)
        throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                    pricing library");

    val *= fx_spot * NumberOfUnits_;

    return val;
}


protected override void Validate()
{

    Pre_string_ = Name() + "  (" + Reference_ + ")";

    if (Time_To_Expiry_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than zero
                                                                        years");
    }

    if (NumberOfUnits_ < 0)
    {
        throw new Exception(Pre_string_ + "--- Number of units cannot be less than
                                                                        zero");
    }

    if (Strike_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
    }
 }

private double MonteCarloSim(ref int iflag)
{

    double[] C = new double[n_ * n_];
    double half = 0.5;
    double zero = 0.0;
    double sumit_val = zero;
    double tol = 1.0e-8;
    double opt_val = 0.0;

    // set the covariance matrix
    for (int i = 0; i < n_; ++i)
    {
        for (int j = 0; j < n_; ++j) {
            C[i * n_ + j] = Sigma_[i] * Sigma_[j] * Correlations_[i, j]
                            * Time_To_Expiry_;
        }
    }

    double[] MEANS = new double[n_];
    // set the means
    for (int i = 0; i < n_; ++i) {
       MEANS[i] = (RiskFreeRate_ - Sigma_[i] * Sigma_[i] * half) * Time_To_Expiry_;
    }

    int seed = 111;

    PricingUtils.set_seed(seed);

    int len_rvec = ((n_ + 1) * (n_ + 2)) / 2 + 1;
    double[] rvec = new double[len_rvec];
    double[] Z = new double[n_];

    double disc = Math.Exp(-RiskFreeRate_ * Time_To_Expiry_);
```

Code excerpt 8.12 (*Continued*).

```
            int is_fcall = 1;
            PricingUtils.multivariate_normal(is_fcall, ref MEANS[0], n_, ref C[0], n_, tol,
                                                        ref rvec[0], ref Z[0], ref iflag);
            is_fcall = 0;
            for (int i = 1; i <= NumberScenarios_; ++i)
            {
               PricingUtils.multivariate_normal(is_fcall, ref MEANS[0], n_, ref C[0], n_, tol,
                                                         ref rvec[0], ref Z[0], ref iflag);
                for (int jj=0; jj < n_; ++jj) {
                   ST_[jj] = S_[jj] * Math.Exp(Z[jj]);
                }
                sumit_val += Payoff();
            }
            opt_val = sumit_val * disc / (double)NumberScenarios_;

            return opt_val;
        }

    }
}
```

Code excerpt 8.12 (*Continued*).

tion on an arbitrary number of underlying assets; the derived class also must implement the abstract method `Payoff`. In the earlier sections of this book, we have considered options with standard payoffs such as: vanilla put, vanilla call, and call/put on the min/max of a number of assets. However, the class `GenericEquityBasketOptionDeal` now opens the possibility of supplying a user-defined `Payoff` function so that options with nonstandard payoffs can be valued.

Below we provide some example results for options on four and ten assets. The assets were: `Drinks-4U`, `Beverage-Ltd`, `H2O-Ltd`, and `Fine-Wines-Ltd`. The trade attributes are a time to expiry of one year, all volatilities are 0.2, and the number of units is 100. Other information required to price the option, such as the correlations between the equities and the risk free interest rate, is taken from the market data dictionaries.

The syntax for using the deal class `GenericEquityBasketOptionDeal` with the portfolio definition File is:

```
Trade=GenericEquityBasketOptionDeal:Payoff_MaxPut,Reference=1A,Strike=100.0,_
Volatilities=0.2%0.2%0.2%0.2,_
Equities=Drinks-4U%Beverage-Ltd%H2O-Ltd%Fine-Wines-Ltd,NumberOfUnits=100,_
Time_To_Expiry=1.0,NumberScenarios=1000
```

while that for calling the deal class `FourEquityOptionDeal` is:

```
Trade=FourEquityOptionDeal,Reference=1B,Volatility1=0.2,Volatility2=0.2,_
Volatility3=0.2,Volatility4=0.2,Equity1=Drinks-4U,Equity2=Beverage-Ltd,_
Equity3=H2O-Ltd,Equity4=Fine-Wines-Ltd,NumberOfUnits=100,Strike=100.0,_
Time_To_Expiry=1.0,OptionType=Put,MinMax=Maximum,MonteCarlo=Yes,NumberScenarios=1000
```

The crucial diffference is that the entry for `GenericEquityBasketOption-Deal` contains the extra directive `Trade=GenericEquityBasketOptionDeal: Payoff_MaxPut`, whereas `FourEquityOptionDeal` is the usual `Trade= FourEquityOptionDeal`. The directive `GenericEquityBasketOption-Deal:Payoff_MaxPut` means that the contents of the file `Payoff_MaxPut.`

txt will be compiled at runtime and thereby create the (sub)class GenericEquityBasketOptionDeal_MaxPut, which is derived from the abstract base class GenericEquityBasketOptionDeal. The .NET assembly containing the class GenericEquityBasketOptionDeal_MaxPut is stored in memory and its Price() method is called to value the option.

The file Payoff_MaxPut.txt contains the following C# code:

```
using System;
namespace Computational_Lib
{
    public class GenericEquityBasketOptionDeal_MaxPut : GenericEquityBasketOptionDeal
    {
       public override string Name()
        {
            string temp_string = "";
            temp_string = "Generic option: Put on the maximum of " + n_.ToString() + " assets";
            return temp_string;
        }

        public override double Payoff() { // implement max, put
           double the_max = 0.0;
           double pay_val = 0.0;
           double zero = 0.0;

           the_max = ST_[0];
           for (int jj = 1; jj < n_; ++jj)
           {
              if (ST_[jj] > the_max) the_max = ST_[jj];
           }
           pay_val = Math.Max(Strike_ - the_max, zero);

           return pay_val;
        }
    }
}
```

In the above code ST_, Strike_, and n_ are data members of the base class GenericEquityBasketOptionDeal. We now present some other entries in the portfolio definition file which illustrate the versatility of the deal class GenericEquityBasketOptionDeal.

```
// Call on average of 4 assets
Trade=GenericEquityBasketOptionDeal:Payoff_AvgCall,Reference=5,Strike=100.0,_
Volatilities=0.2%0.2%0.2%0.2,Equities=Drinks-4U%Beverage-Ltd%H2O-Ltd%Fine-Wines-Ltd,_
NumberOfUnits=100,Time_To_Expiry=1.0,NumberScenarios=1000

// Put on the average of 10 assets (Strike=100)
Trade=GenericEquityBasketOptionDeal:Payoff_AvgPut,Reference=8,Strike=100.0,_
Volatilities=0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2,_
Equities=Drinks-4U%Beverage-Ltd%H2O-Ltd%Fine-Wines-Ltd%The-English-Beer-Company_
%Water-Works-Ltd%Welsh-Spring%ThamesBeer_
%Edingburgh-Whiskey%The-Wine-Box,NumberOfUnits=100,Time_To_Expiry=1.0,NumberScenarios=10000

// Put on the average of 10 assets (Strike=99)
Trade=GenericEquityBasketOptionDeal:Payoff_AvgPut,Reference=9,Strike=99.0,_
Volatilities=0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2%0.2,_
Equities=Drinks-4U%Beverage-Ltd%H2O-Ltd%Fine-Wines-Ltd%The-English-Beer-Company_
%Water-Works-Ltd%Welsh-Spring%ThamesBeer%_
Edingburgh-Whiskey%The-Wine-Box,NumberOfUnits=100,Time_To_Expiry=1.0,NumberScenarios=10000
```

where the file Payoff_AvgCall.txt contains the C# code:

```
using System;

namespace Computational_Lib
{
    public class GenericEquityBasketOptionDealAverageCall : GenericEquityBasketOptionDeal
    {
        public override string Name()
            string temp_string = "";
            temp_string = "Generic option: Call on the average of_
             " + n_.ToString() + " assets";
            return temp_string;
```

```
        }

        public override double Payoff() {  // implement Call on average of n_assets
           double the_average = 0.0;
           double pay_val = 0.0;
           double zero = 0.0;
           the_average = ST_[0];
           for (int jj = 1; jj < n_; ++jj)
           {
              the_average += ST_[jj];
           }
           the_average = the_average/n_;

           pay_val = Math.Max(the_average - Strike_, zero);

           return pay_val;
        }
    }
}
```

The contents of the file `Payoff_AvgPut` can be deduced from `Payoff_AvgCall` in the obvious manner.

The output from the application is given below:

```
============================================================
TestGenericEQ in units of GBP
TestGenericEQ   :31/07/2007 19:05:10
============================
 23.0100=1A,Generic option: Put on the maximum of 4 assets
 23.0100=1B,Four Equity Option
             .    .    .
 681.4034=5,Generic option: Call on the average of 4 assets
            .    .    .
 338.3212=8,Generic option: Put on the average of 10 assets
 302.6056=9,Generic option: Put on the average of 10 assets
=========================
TOTAL VALUE =  10936.18 GBP
=========================
```

It can be seen that result 1A, obtained using the deal class `GenericEquity-BasketOptionDeal`, is exactly the same as result 1B, which was computed with the deal class `FourEquityOptionDeal`. This is because in both cases Monte Carlo simulation is used, and the same initial random seed is used for all Monte Carlo simulations.

8.4.4 *Equity barrier option*

Code excerpt 8.13 gives the C# code for computing the value of an equity barrier option.

```
using System;

namespace Computational_Lib
{
    public class DownOutEquityOptionDeal : BaseDeal
    {
        public string Equity { get { return EquityName_; } set { EquityName_ = value; } }
        public double Barrier_Level { get { return BarrierLevel_; }_
           set { BarrierLevel_ = value; } }
        public double Time_To_Expiry { get { return Time_To_Expiry_; }_
           set { Time_To_Expiry_ = value; } }
        public PutCall OptionType { get { return OptionType_; } set { OptionType_ = value; } }
        public double Strike { get { return Strike_; } set { Strike_ = value; } }
        public CalculationMethod CalcMethod { get { return CalcMethod_; }_
           set { CalcMethod_ = value; } }
```

Code excerpt 8.13 C# code to compute the value of an equity barrier option.

```
public EuropeanAmerican ExerciseStyle { get { return ExerciseStyle_; }_
   set { ExerciseStyle_ = value; } }
public int TimeSteps { get { return TimeSteps_; } set { TimeSteps_ = value; } }
public int NumberScenarios { get { return NumberScenarios_; }_
   set { NumberScenarios_ = value; } }
public bool UseBrownianBridge{ get { return UseBrownianBridge_; }_
   set { UseBrownianBridge_ = value; } }
public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }
public double Volatility { get { return Sigma1_; } set { Sigma1_ = value; } }
public int NumberOfUnits { get { return NumberOfUnits_; }_
   set { NumberOfUnits_ = value; } }

protected PutCall OptionType_ = PutCall.Call;
protected double Strike_ = 0;
protected double BarrierLevel_ = 0.0;
protected string EquityName_ = "";
protected double Time_To_Expiry_ = 0.0;
protected double RiskFreeRate_ = 0.0;
protected double DividendYield_ = 0.0;
protected string Pre_string_ = "";
protected CalculationMethod CalcMethod_ = CalculationMethod.Analytic;
protected int TimeSteps_ = 300;
protected int NumberScenarios_ = 3000;
protected bool UseBrownianBridge_ = true;
protected double S1_ = 0.0;
protected double Sigma1_ = 0.0;
protected int NumberOfUnits_ = 1;
protected BuySell BuySell_ = BuySell.Buy;
protected EuropeanAmerican ExerciseStyle_ = EuropeanAmerican.European;
protected string Currency_ = "";

public override string Name()
{
    return "Down Out Equity Option";
}

public override double Price()
{
    Validate();
    double val = 0.0;
    double fx_spot = 0.0;
    try
    {
        S1_ = MarketDataDictionaries.EquityTable[EquityName_].Spot;
        Currency_ = MarketDataDictionaries.EquityTable[EquityName_].Currency;
        // get equity volatility (assumed constant)
        DividendYield_ = MarketDataDictionaries.EquityTable[EquityName_].DivYield;
        // get equity dividend yield
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                              + EquityName_);
    }

    double discount_fac = 0.0;
    try
    {
        ICurve DF = MarketDataDictionaries.CurrencyTable[Currency_].YieldCurve;
        // obtain the discount factor
        discount_fac = DF[0, Time_To_Expiry_];
        RiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
        fx_spot = MarketDataDictionaries.CurrencyTable[Currency_].spot;
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                                + Currency_);
    }

    int iflag, put, is_american;

    iflag = 0;
    put = OptionType_ == PutCall.Put ? 1 : 0;
    is_american = ExerciseStyle_ == EuropeanAmerican.American ? 1 : 0;
```

Code excerpt 8.13 (*Continued*).

```
        BarrierLevel_ = Math.Max(BarrierLevel_, PricingUtils.EPS);

        if (CalcMethod_ == CalculationMethod.Analytic)
        {
            if (put == 1) throw new Exception(Pre_string_ + "--- Can't price a put using
                                                              this calculation method");
            if (is_american == 1) throw new Exception(Pre_string_ + "--- Can't price an
                                        American option using this calculation method");

            PricingUtils.bs_opt_barrier_downout_call(ref val, BarrierLevel_,
                                S1_, Strike_, Sigma1_,
                                Time_To_Expiry_, RiskFreeRate_, DividendYield_,
                                                                    ref iflag);
            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                  pricing library");
        }
        else if (CalcMethod_ == CalculationMethod.Numeric)
        {
            if (put == 1) throw new Exception(Pre_string_ + "--- Can't price a put using
                                                              this calculation method");

            int n_sigma = 2;

            // set up the parameters so that have "reasonable accuracy"
            double[] sigma_array = new double[n_sigma];
            double[] sigma_times = new double[n_sigma];

            sigma_array[0] = Sigma1_;
            sigma_array[1] = Sigma1_;

            sigma_times[0] = 0.0;
            sigma_times[1] = Time_To_Expiry_;

            int nt = 100;
            int ns_below_S0 = nt / 2;
            int ns_above_S0 = nt / 2;
            double theta_m = 0.5;
            double UpperBarrierLevel = S1_ * 5.0;

            iflag = 0;

            PricingUtils.dko_call(BarrierLevel_, UpperBarrierLevel,
                     theta_m, S1_, ref sigma_array[0], ref sigma_times[0],
                     n_sigma, RiskFreeRate_, Time_To_Expiry_,
                     Strike_, is_american, ref val,
                     IntPtr.Zero, DividendYield_, ns_below_S0, ns_above_S0,
                     nt, ref iflag);

            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                  pricing library");
        }
        else
        {
            bool is_put = (put == 1);
            if (is_american == 1) throw new Exception(Pre_string_ + "--- Can't price an
                                        American option using this calculation method");

            if (S1_ < BarrierLevel_)  // the opton is already knocked out
               val = 0.0;
            else
               val = MonteCarloSim(is_put);
        }

        val *= fx_spot * NumberOfUnits_;

        return val;
    }
```

Code excerpt 8.13 (*Continued*).

```
protected override void Validate()
{
    Pre_string_ = Name() + "  (" + Reference_ + ")";

    if (NumberOfUnits_ < 0)
    {
        throw new Exception(Pre_string_ + "--- Number of units cannot be less than
                                                                                zero");
    }
    if (Time_To_Expiry_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than zero
                                                                                years");
    }
    if (RiskFreeRate_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- Risk free rate cannot be less than
                                                                               zero");
    }
    if (Strike_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
    }
    if (BarrierLevel_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- BarrierLevel cannot be less than zero");
    }
    if (Sigma1_ < 0.0)
    {
        throw new Exception(Pre_string_ + "--- Volatility cannot be less than zero");
    }
}


private double MonteCarloSim(bool is_put)
{
    // Use the Brownian Bridge to compute the value of a down and out call option

    int seed = 111;
    double[] asset_path = new double[TimeSteps_];
    double time_step = Time_To_Expiry_ / TimeSteps_;
    double sqrt_time_step = System.Math.Sqrt(time_step);
    double disc = System.Math.Exp(-RiskFreeRate_ * Time_To_Expiry_);

    PricingUtils.set_seed(seed);

    double opt_val = 0.0;
    bool not_out = true;
    int k = 0;
    double STN = 0.0;
    double mean = (RiskFreeRate_ - DividendYield_ - Sigma1_ * Sigma1_ * 0.5)
                  * time_step;
    double std = System.Math.Sqrt(Sigma1_ * Sigma1_ * time_step);
    double z;
    double sum_opt_vals = 0.0;

    for (int i = 0; i < NumberScenarios_; ++i)
    {
        // generate the asset path
        double ST1 = S1_;
        not_out = true;
        k = 0;

        while (not_out && k < TimeSteps_)
        {
            z = PricingUtils.RndNorm(mean, std);
            STN = ST1 * System.Math.Exp(z);
            if (STN < BarrierLevel_) not_out = false;
            ST1 = STN;
            asset_path[k] = STN;
            ++k;
        }
```

Code excerpt 8.13 (*Continued*).

```
                if (is_put)
                {
                    opt_val = System.Math.Max(Strike_ - STN, 0.0);
                }
                else
                {
                    opt_val = System.Math.Max(STN - Strike_, 0.0);
                }

                if (not_out)
                { // only has value if asset value is above the barrier_level
                    // compute the probability that the asset remained above the barrier
                    if (UseBrownianBridge)
                    {
                        double total_probability_above = 1.0, pr;
                        double sigma_2 = Sigma1_ * Sigma1_;
                        double log_barrier_level = System.Math.Log(BarrierLevel_);
                        double fac;
                        for (int jj = 0; jj < TimeSteps_ - 1; ++jj)
                        {
                            double log_S_i = System.Math.Log(asset_path[jj]);
                            double log_S_i1 = System.Math.Log(asset_path[jj + 1]);
                            fac = 2.0 * (log_barrier_level - log_S_i)
                                  * (log_barrier_level - log_S_i1) / (sigma_2 * time_step);
                            pr = (1.0 - System.Math.Exp(-fac));
                            // probability of staying above the barrier between i and i+1
                            total_probability_above *= pr;
                        }
                        sum_opt_vals += total_probability_above * opt_val * disc;
                    }
                    else
                    { // don't use the Brownian Bridge
                        sum_opt_vals += opt_val * disc;
                    }
                }
            }
            double temp = sum_opt_vals / (double)NumberScenarios_;

            return temp;
        }
    }
}
```

Code excerpt 8.13 (*Continued*).

Below we show the results of using the deal class `DownOutEquityOption-Deal` to value Down and Out call options on `LaserComm` which is a GBP equity with current (spot) price of £95, and a dividend yield of 5 percent (i.e., 0.05). All the options priced had a barrier level of £90, a strike of £90, a time to expiry of one year, and a volatility of 20 percent (i.e., 0.2). The first value, £3.8347, was computed by a call to `bs_opt_barrier_downout_call`, which uses the closed form analytic expression provided in Code excerpt 2.6.

The second price, £3.8269, which is in close agreement with the first, was obtained from `dko_call` and uses a finite-difference grid. The third valuation was also computed using `dko_call`, and illustrates the early exercise premium for an American call option (with a nonzero dividend). The other values were estimated using Monte Carlo simulation as the number of scenarios varied from 1000 to 64000; the default of 300 time steps was used throughout.

It can be seen that, when the Brownian bridge is used, much closer agreement is obtained with both the analytic and numeric estimates.

```
DownOutTests in units of GBP
DownOutTests   :26/07/2007 13:11:28
============================
 3.8347=Analytic,Down Out Equity Option
```

```
 3.8269=Numeric,Down Out Equity Option
 3.8860=Numeric (American style),Down Out Equity Option
 4.1871=MonteCarlo(1000 Scenarios: not using BrownianBridge),Down Out Equity Option
 3.8908=MonteCarlo(2000 Scenarios: not using BrownianBridge),Down Out Equity Option
 4.1968=MonteCarlo(4000 Scenarios: not using BrownianBridge),Down Out Equity Option
 4.1176=MonteCarlo(8000 Scenarios: not using BrownianBridge),Down Out Equity Option
 4.1790=MonteCarlo(16000 Scenarios: not using BrownianBridge),Down Out Equity Option
 4.1961=MonteCarlo(32000 Scenarios: not using BrownianBridge),Down Out Equity Option
 4.1833=MonteCarlo(64000 Scenarios: not using BrownianBridge),Down Out Equity Option
 3.8375=MonteCarlo(1000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.5469=MonteCarlo(2000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.8737=MonteCarlo(4000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.7356=MonteCarlo(8000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.8089=MonteCarlo(16000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.8506=MonteCarlo(32000 Scenarios: using BrownianBridge),Down Out Equity Option
 3.8482=MonteCarlo(64000 Scenarios: using BrownianBridge),Down Out Equity Option
========================
TOTAL VALUE =  70.83 GBP
========================
```

8.5 FX deal classes

Here we provide code for valuing FX derivatives. The FX option routines are very similar to the equity option routines we have already considered, the fundamental difference being that for FX routines there is both a domestic and foreign currency. The FX routine calls the Black–Scholes routine with the dividend yield set to the foreign currency risk free interest rate, and the supplied volatility is that of the foreign/domestic exchange rate. In the market data file the currency FX spot rates are with respect to the base currency.

8.5.1 FX forward

Code excerpt 8.14 gives the C# code to compute the value of FX forwards.

```
using System;

namespace Computational_Lib
{

    public class FXForwardDeal : BaseDeal
    {
        public double ForeignAmount { get { return fForeignAmount; }_
           set { fForeignAmount = value; } }

        // Note: Strike is the number of units of domestic currency required to
        //       obtain one unit of foreign currency.
        public double Strike { get { return fStrike; } set { fStrike = value; } }
        public string ForeignCurrency { get { return fForeignCurrency; }_
           set { fForeignCurrency = value; } }
        public string DomesticCurrency { get { return fDomesticCurrency; }_
           set { fDomesticCurrency = value; } }
        public BuySell BuySell { get { return fBuySell; } set { fBuySell = value; } }
        public double Settlement { get { return fSettlement; } set { fSettlement = value; } }
```

Code excerpt 8.14 C# code to compute the value of FX forwards.

```
        protected double fStrike    = 0;
        protected string fForeignCurrency = "";
        protected string fDomesticCurrency = "";
        protected double fForeignAmount = 0;
        protected BuySell fBuySell = BuySell.Buy;
        protected double fSettlement = 0;
        protected string pre_string = "";

        public override string  Name()
        {
            return "FX Forward";
        }

        public override double Price()
        {
            double val=0.0;
            Validate();
            double sign = fBuySell == BuySell.Buy ? 1.0 : -1.0;

            try
            {
                ICurve DF_F = CurrencyTable[fForeignCurrency].YieldCurve;
                // obtain the discount factor
                ICurve DF_D = CurrencyTable[fDomesticCurrency].YieldCurve;
                // obtain the discount factor
                double X_fb = CurrencyTable[fForeignCurrency].spot;
                double X_db = CurrencyTable[fDomesticCurrency].spot;
                double DF_f = DF_F[0,fSettlement];
                double DF_d = DF_D[0,fSettlement];

                val = fForeignAmount *( DF_f * X_fb - X_db * DF_d * fStrike);

                val = val * sign;
            }
            catch(Exception ex)
            {
                throw new Exception(pre_string + " : " + ex.Message);
            }

            return val;
        }

        protected override void Validate()
        {
            pre_string = Name() + "  (" + fReference + ")";

        }
    }
}
```

Code excerpt 8.14 C# code to compute the value of FX forwards.

8.5.2 Single FX option

The code for the single FX option, given in Code excerpt 8.15, is very similar to that for the single equity option. For example, European equity options are priced using the call:

```
PricingUtils.black_scholes(ref val, IntPtr.Zero, s0, Strike_, Sigma_,
Time_To_Expiry_, RiskFreeRate, DividendYield_, put,
ref iflag);
```

while European FX options use:

```
PricingUtils.black_scholes(ref val, IntPtr.Zero, S0, Strike_b,Sigma_f_d_,
Time_To_Expiry_, DomesticRiskFreeRate_, ForeignRiskFreeRate_, put, ref iflag);
```

```
using System;

namespace Computational_Lib
{

    public class FXOptionDeal: BaseDeal
    {

        public int NumberOfUnits { get { return NumberOfUnits_; }_
           set { NumberOfUnits_ = value; } }
        // Note: Strike is the number of units of domestic currency required to
        //       obtain one unit of foreign currency.
        public double Strike { get { return Strike_f_d_; } set { Strike_f_d_ = value; } }

// Volatility is that of the Foreign/Domestic exchange rate.
        public double Volatility { get { return Sigma_f_d_; } set { Sigma_f_d_ = value; } }

        public string ForeignCurrency { get { return ForeignCurrency_; }_
           set { ForeignCurrency_ = value; } }
        public string DomesticCurrency { get { return DomesticCurrency_; }_
           set { DomesticCurrency_ = value; } }
        public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }
        public double Time_To_Expiry { get { return Time_To_Expiry_; }_
           set { Time_To_Expiry_ = value; } }
        public PutCall OptionType { get { return OptionType_; } set { OptionType_ = value; } }
        public EuropeanAmerican ExerciseStyle { get { return ExerciseStyle_; }_
           set { ExerciseStyle_ = value; } }
        protected double Strike_f_d_ = 0.0;
        protected string ForeignCurrency_ = "";
        protected string DomesticCurrency_ = "";
        protected BuySell BuySell_ = BuySell.Buy;
        protected int NumberOfUnits_ = 1;
        protected PutCall OptionType_ = PutCall.Put;
        protected EuropeanAmerican ExerciseStyle_ = EuropeanAmerican.European;
        protected double Time_To_Expiry_ = 0.0;
        protected double ForeignRiskFreeRate_ = 0.0;
        protected double DomesticRiskFreeRate_ = 0.0;
        protected double Sigma_f_d_ = 0.0;
        protected string Pre_string_ = "";

        public override string Name()
        {
            return "FX Option";
        }

        public override double Price()
        {

            Validate();

            double val = 0.0;
            double[] greeks = new double[6];

            int iflag, put;
            double discount_fac = 0.0;
            double X_f_b = 0.0, X_d_b = 0.0;
            double S0=0.0,Strike_b;

            // Get domestic currency information
            try
            {
                ICurve DF = MarketDataDictionaries.CurrencyTable[DomesticCurrency_].YieldCurve;
                // obtain the domestic discount factor
                discount_fac = DF[0, Time_To_Expiry_];
                DomesticRiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
                X_d_b = MarketDataDictionaries.CurrencyTable[DomesticCurrency_].spot;
                Strike_b = X_d_b * Strike_f_d_;
                // Strike_b is the Strike in base currency units
            }
            catch
            {
                throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                                + DomesticCurrency_);
            }
```

Code excerpt 8.15 C# code to compute the value of FX options.

```
        // Get foreign currency information
        try
        {
            ICurve DF = MarketDataDictionaries.CurrencyTable[ForeignCurrency_].YieldCurve;
            // obtain the domestic discount factor
            discount_fac = DF[0, Time_To_Expiry_];
            ForeignRiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
            X_f_b = MarketDataDictionaries.CurrencyTable[ForeignCurrency_].spot;
            S0 = X_f_b;  // Foreign exchange wrt base currency
        }
        catch
        {
            throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                              + ForeignCurrency_);
        }

        iflag = 0;
        put = OptionType_ == PutCall.Put ? 1 : 0;

        if (ExerciseStyle_ == EuropeanAmerican.European)  // use BlackScholes
        {
            // Note: A null pointer is supplied so that the Greeks are not computed
            // Dividend yield is set to foreign risk free rate
            // Risk free interest rate is set to the domestic rate
            // S0 the value of the "asset" in base cuurency units
            // val is  the value of the FX option in base currency units
            PricingUtils.black_scholes(ref val, IntPtr.Zero, S0, Strike_b, Sigma_f_d_,
                            Time_To_Expiry_,
                            DomesticRiskFreeRate_, ForeignRiskFreeRate_, put, ref iflag);


        }
        else
        {  // Use Finite Difference Grid - Crank Nicolson
            double theta = 0.5;
            int is_american = 1;

            // fix the geometry of the grid (these avluse should give "reasonable" results)
            int ns = 50;  // 50 divisions on asset axis
            int nt = 50;  // 50 divisions on time axis
            double smax = 10.0 * S0;

            PricingUtils.opt_gfd(theta, S0, Sigma_f_d_, DomesticRiskFreeRate_,
                  Time_To_Expiry_, Strike,
                  is_american, put, ref val, IntPtr.Zero, ForeignRiskFreeRate_, ns,
                  nt, smax, ref iflag);

            // val is  the value of the FX option in base currency units
        }

        if (iflag != 0)
            throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                  pricing library");

        val *= NumberOfUnits_;

        return val;
    }

    protected override void Validate()
    {

        Pre_string_ = Name() + "  (" + Reference_ + ")";
        if (Time_To_Expiry_ < 0.0)
        {
            throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than zero
                                                                              years");
        }
        if (NumberOfUnits_ < 0)
        {
            throw new Exception(Pre_string_ + "--- Number of units cannot be less than
                                                                              zero");
        }
```

Code excerpt 8.15 (*Continued*).

```
            if (Strike_f_d_ < 0.0)
            {
                throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
            }
        }
    }
}
```

Code excerpt 8.15 (*Continued*).

It can be seen that, when pricing FX options, the foreign risk free rate is used instead of the dividend yield, and the supplied volatility is that of the foreign/domestic exchange rate. Another difference is that the equity option value `val` returned by the call to `black_scholes` is in domestic currency units, and is then converted to base currency, while in the case of FX options the value `val` is already in base currency units, and requires no conversion.

8.5.3 FX barrier option

The C# code for the Barrier option is given in Code excerpt 8.16.

```
using System;

namespace Computational_Lib
{
    public class DownOutFXOptionDeal: BaseDeal
    {
        public int NumberOfUnits { get { return NumberOfUnits_; }_
           set { NumberOfUnits_ = value; } }
        // Note: Strike is the number of units of domestic currency required to obtain one
        //        unit of foreign currency.
        public double Strike { get { return Strike_f_d_; } set { Strike_f_d_ = value; } }
        // Barrier is in the same units a the strike
        public double Barrier_Level { get { return Barrier_f_d_; }_
           set { Barrier_f_d_ = value; } }
// Volatiliy is that of the Foreign/Domestic exchange rate.
        public double Volatility { get { return Sigma_f_d_; } set { Sigma_f_d_ = value; } }
        public string ForeignCurrency { get { return ForeignCurrency_; }_
           set { ForeignCurrency_ = value; } }
        public string DomesticCurrency { get { return DomesticCurrency_; }_
           set { DomesticCurrency_ = value; } }
        public BuySell BuySell { get { return BuySell_; } set { BuySell_ = value; } }

        public CalculationMethod CalcMethod { get { return CalcMethod_; }_
           set { CalcMethod_ = value; } }
        public EuropeanAmerican ExerciseStyle { get { return ExerciseStyle_; }_
           set { ExerciseStyle_ = value; } }
        public int NumberScenarios { get { return NumberScenarios_; }_
           set { NumberScenarios_ = value; } }
        public bool UseBrownianBridge { get { return UseBrownianBridge_; }_
           set { UseBrownianBridge_ = value; } }

        protected double Strike_f_d_ = 0.0;
        protected double Barrier_f_d_ = 0.0;
        protected string ForeignCurrency_ = "";
        protected string DomesticCurrency_ = "";
        protected BuySell BuySell_ = BuySell.Buy;
        protected int NumberOfUnits_ = 1;
        public double Time_To_Expiry { get { return Time_To_Expiry_; }_
           set { Time_To_Expiry_ = value; } }
        public PutCall OptionType { get { return OptionType; } set { OptionType_ = value; } }
        protected PutCall OptionType_ = PutCall.Call;
        protected double Time_To_Expiry_ = 0.0;
```

Code excerpt 8.16 C# code to compute the value of FX barrier options.

```
protected double ForeignRiskFreeRate_ = 0.0;
protected double DomesticRiskFreeRate_ = 0.0;
protected double Sigma_f_d_ = 0.0;
protected EuropeanAmerican ExerciseStyle_ = EuropeanAmerican.European;
protected CalculationMethod CalcMethod_ = CalculationMethod.Analytic;
protected int TimeSteps_ = 300;
protected int NumberScenarios_ = 3000;
protected double S0_,Strike_b_,BarrierLevel_b_;
protected bool UseBrownianBridge_ = true;
protected string Pre_string_ = "";

public override string Name()
{
    return "Down Out FX Option";
}

public override double Price()
{
    Validate();

    double val = 0.0;
    int iflag, put, is_american;
    double discount_fac = 0.0;
    double X_f_b = 0.0, X_d_b = 0.0;

    // Get domestic currency information
    try
    {
        ICurve DF = MarketDataDictionaries.CurrencyTable[DomesticCurrency_].YieldCurve;
        // obtain the domestic discount factor
        discount_fac = DF[0, Time_To_Expiry_];
        DomesticRiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
        X_d_b = MarketDataDictionaries.CurrencyTable[DomesticCurrency_].spot;
        Strike_b_ = X_d_b * Strike_f_d_;
        // Strike is the Strike in base currency units
        BarrierLevel_b_ = X_d_b * Barrier_f_d_;
        BarrierLevel_b_ = Math.Max(BarrierLevel_b_, PricingUtils.EPS);
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                    + DomesticCurrency_);
    }

    // Get foreign currency information
    try
    {
        ICurve DF = MarketDataDictionaries.CurrencyTable[ForeignCurrency_].YieldCurve;
        // obtain the domestic discount factor
        discount_fac = DF[0, Time_To_Expiry_];
        ForeignRiskFreeRate_ = -Math.Log(discount_fac) / Time_To_Expiry_;
        X_f_b = MarketDataDictionaries.CurrencyTable[ForeignCurrency_].spot;
        S0_ = X_f_b;  // Foreign exchange wrt base currency
    }
    catch
    {
        throw new Exception(Pre_string_ + "--- No Market Data supplied for "
                                                     + ForeignCurrency_);
    }

    iflag = 0;
    put = OptionType_ == PutCall.Put ? 1 : 0;
    is_american = ExerciseStyle_ == EuropeanAmerican.American ? 1 : 0;

    if (CalcMethod_ == CalculationMethod.Analytic)
    {
        if (put == 1) throw new Exception(Pre_string_ + "--- Can't price a put using
                                                        this calculation method");
        if (is_american == 1) throw new Exception(Pre_string_ + "--- Can't price an
                                    American option using this calculation method");

        // call C routine.
        // Note: A null pointer is supplied so that the Greeks are not computed
        // Dividend yield is set to foreign risk free rate
        // Risk free interest rate is set to the domestic rate
```

Code excerpt 8.16 (*Continued*).

```
            // val is  the value of the FX option in base currency units
           PricingUtils.bs_opt_barrier_downout_call(ref val, BarrierLevel_b_,
                                S0_, Strike_b_, Sigma_f_d_,
                                Time_To_Expiry_, DomesticRiskFreeRate_,
                                ForeignRiskFreeRate_, ref iflag);

            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                        pricing library");
        }
        else if (CalcMethod_ == CalculationMethod.Numeric)
        {
            if (put == 1) throw new Exception(Pre_string_ + "--- Can't price a put using
                                                        this calculation method");

            int n_sigma = 2;

            // set up the parameters so that have "reasonable accuracy"
            double[] sigma_array = new double[n_sigma];
            double[] sigma_times = new double[n_sigma];

            sigma_array[0] = Sigma_f_d_;
            sigma_array[1] = Sigma_f_d_;
            sigma_times[0] = 0.0;
            sigma_times[1] = Time_To_Expiry_;

            int nt = 100;
            int ns_below_S0 = nt / 2;
            int ns_above_S0 = nt / 2;
            double theta_m = 0.5;
            double UpperBarrierLevel = S0_ * 5.0;

            iflag = 0;
            // val is  the value of the FX option in base currency units
            PricingUtils.dko_call(BarrierLevel_b_, UpperBarrierLevel,
                     theta_m, S0_, ref sigma_array[0], ref sigma_times[0],
                     n_sigma, DomesticRiskFreeRate_, Time_To_Expiry_,
                     Strike_b_, is_american, ref val,
                     IntPtr.Zero, ForeignRiskFreeRate_, ns_below_S0, ns_above_S0,
                     nt, ref iflag);

            if (iflag != 0)
                throw new Exception(Pre_string_ + "--- An error occurred in a call to the
                                                                        pricing library");
        }
        else
        {
            bool is_put = (put == 1);
            if (is_american == 1) throw new Exception(Pre_string_ + "--- Can't price an
                                        American option using this calculation method");

            if (S0_ < BarrierLevel_b) // the option has already been knocked out
               val = 0.0;
            else
               val = MonteCarloSim(is_put);

        }


         val *= NumberOfUnits_;

        return val;
    }

    protected override void Validate()
    {

        Pre_string_ = Name() + "  (" + Reference_ + ")";
        if (Time_To_Expiry_ < 0.0)
        {
            throw new Exception(Pre_string_ + "--- Time to expiry cannot be less than
                                                                          zero years");
        }
```

Code excerpt 8.16 (*Continued*).

```
        if (NumberOfUnits_ < 0)
        {
            throw new Exception(Pre_string_ + "--- Number of units cannot be less than
                                                                                zero");
        }
        if (Strike_f_d_ < 0.0)
        {
            throw new Exception(Pre_string_ + "--- The strike cannot be less than zero");
        }
        if (Barrier_f_d_ < 0.0)
        {
            throw new Exception(Pre_string_ + "--- BarrierLevel cannot be less than zero");
        }
        if (Sigma_f_d_ < 0.0)
        {
            throw new Exception(Pre_string_ + "--- Volatility cannot be less than zero");
        }
    }

    private double MonteCarloSim(bool is_put)
    {
        // Use the Brownian Bridge to compute the value of a down and out call option

        int seed = 111;
        double[] asset_path = new double[TimeSteps_];
        double time_step = Time_To_Expiry_ / TimeSteps_;
        double sqrt_time_step = System.Math.Sqrt(time_step);
        double disc = System.Math.Exp(-DomesticRiskFreeRate_ * Time_To_Expiry_);

        PricingUtils.set_seed(seed);

        double opt_val = 0.0;
        bool not_out = true;
        int k = 0;
        double STN = 0.0;
        double mean = (DomesticRiskFreeRate_ - ForeignRiskFreeRate_
                    - Sigma_f_d_ * Sigma_f_d_ * 0.5) * time_step;
        double std = System.Math.Sqrt(Sigma_f_d_ * Sigma_f_d_ * time_step);
        double z;
        double sum_opt_vals = 0.0;

        for (int i = 0; i < NumberScenarios_; ++i)
        {
            // generate the asset path
            double ST1 = S0_;
            not_out = true;
            k = 0;

            while (not_out && k < TimeSteps_)
            {
                z = PricingUtils.RndNorm(mean, std);
                STN = ST1 * System.Math.Exp(z);
                if (STN < BarrierLevel_b_) not_out = false;
                ST1 = STN;
                asset_path[k] = STN;
                ++k;
            }
            if (is_put)
            {
                opt_val = System.Math.Max(Strike_b_ - STN, 0.0);
            }
            else
            {
                opt_val = System.Math.Max(STN - Strike_b_, 0.0);
            }

            if (not_out)
            { // only has value if asset value is above the barrier_level
                // compute the probability that the asset remained above the barrier
                if (UseBrownianBridge)
                {
                    double total_probability_above = 1.0, pr;
                    double sigma_2 = Sigma_f_d_ * Sigma_f_d_;
```

Code excerpt 8.16 (*Continued*).

```
                        double log_barrier_level = System.Math.Log(BarrierLevel_b_);
                        double fac;
                        for (int jj = 0; jj < TimeSteps_ - 1; ++jj)
                        {
                            double log_S_i = System.Math.Log(asset_path[jj]);
                            double log_S_i1 = System.Math.Log(asset_path[jj + 1]);
                            fac = 2.0 * (log_barrier_level - log_S_i)
                                  * (log_barrier_level - log_S_i1) / (sigma_2 * time_step);
                            pr = (1.0 - System.Math.Exp(-fac));
                            // probability of staying above the barrier between i and i+1
                            total_probability_above *= pr;
                        }
                        sum_opt_vals += total_probability_above * opt_val * disc;
                    }
                    else
                    { // don't use the Brownian Bridge
                        sum_opt_vals += opt_val * disc;
                    }
                }
            }
            double temp = sum_opt_vals / (double)NumberScenarios_;

            return temp;
        }
    }
}
```

Code excerpt 8.16 (*Continued*).

Appendix A:

The Greeks for vanilla European options

A.1 Introduction

In this section we will present some useful results which will be used later on to derive expressions for the Greeks.

A fundamental result of calculus is that:

$$\frac{\partial}{\partial x}\int f(x)\,\mathrm{d}x = f(x) \tag{A.1.1}$$

Also the indefinite integral, $\int f(x)\,\mathrm{d}x$, can be expressed as a definite integral with variable upper bound as follows:

$$\int f(x)\,\mathrm{d}x = \int_a^x f(x)\,\mathrm{d}x + c$$

so

$$\frac{\partial}{\partial x}\int_{y=a}^{y=x} f(y)\,\mathrm{d}y = f(x) \tag{A.1.2}$$

We can now use this result to obtain the derivative of the cumulative distribution function:

$$N_1(x) = \frac{1}{\sqrt{2\pi}}\int_{y=-\infty}^{y=x} \exp\left(-\frac{y^2}{2}\right)\mathrm{d}y$$

which gives:

$$\frac{\partial N_1(x)}{\partial x} = n(x) \tag{A.1.3}$$

where

$$n(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$$

We now derive various results for the parameters d_1 and d_2 which appear in the Black–Scholes equation:

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}} \tag{A.1.4}$$

and

$$d_2 = \frac{\log(S/E) + (r - q - \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t} \tag{A.1.5}$$

We have:

$$\frac{\partial d_2}{\partial S} = \frac{\partial d_1}{\partial S} = \frac{1}{S\sigma\sqrt{T-t}} \tag{A.1.6}$$

$$\frac{\partial d_2}{\partial \sigma} = \frac{\partial d_1}{\partial \sigma} - \sqrt{T-t} \tag{A.1.7}$$

$$\frac{\partial d_1}{\partial r} = \frac{\partial d_2}{\partial r} = \frac{\sqrt{T-t}}{\sigma} \tag{A.1.8}$$

$$\frac{\partial d_2}{\partial t} = \frac{\partial d_1}{\partial t} + \frac{\sigma}{2(T-t)} \tag{A.1.9}$$

Also:

$$\begin{aligned} n(d_2) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{d_2^2}{2}\right) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{d_1^2}{2}\right) \exp\left\{\sigma d_1\sqrt{T-t} - \frac{\sigma^2(T-t)}{2}\right\} \\ &= n(d_1) \exp\left\{\log\left(\frac{S}{E}\right) + \left(r - q + \frac{\sigma^2}{2}\right)(T-t) - \frac{\sigma^2(T-t)}{2}\right\} \end{aligned}$$

so

$$n(d_2) = \frac{S}{E} n(d_1) \exp\big(r(T-t)\big) \exp\big(-q(T-t)\big) \tag{A.1.10}$$

We note that:

$$\frac{\partial N_1(d_1)}{\partial S} = \frac{\partial N_1(d_1)}{\partial d_1}\frac{\partial d_1}{\partial S} = n(d_1)\frac{1}{S\sigma\sqrt{T-t}}$$

This technique will be used for computing the *Greeks*.

A.2 Gamma

Gamma is defined as the second derivative of the option value with respect to the underlying stock price. This means (see Section A.3) it is the rate of change of delta with the underlying stock price.

For a European call the value of gamma is:

$$\Gamma_c = \frac{\partial^2 c}{\partial S^2} = \frac{\partial \Delta_c}{\partial S} = \frac{\partial}{\partial S}\big\{N_1(d_1)\exp\big(-q(T-t)\big)\big\}$$

where the value of Δ_c is given in Section A.3. So

$$\Gamma_c = \exp\big(-q(T-t)\big)\frac{\partial N_1(d_1)}{\partial S} = \exp\big(-q(T-t)\big) n(d_1)\frac{\partial d_1}{\partial S}$$

Therefore:

$$\Gamma_c = \frac{n(d_1)}{S\sigma\sqrt{T-t}}\exp\big(-q(T-t)\big) \tag{A.2.1}$$

The value of gamma for a European put can be calculated similarly:

$$\Gamma_p = \frac{\partial^2 p}{\partial S^2} = \frac{\partial \Delta_p}{\partial S} = \frac{\partial}{\partial S}\big\{\big(N_1(d_1)-1\big)\exp\big(-q(T-t)\big)\big\}$$

where we have used the value of Δ_p, derived in Section A.3. Therefore:

$$\Gamma_p = \exp\big(-q(T-t)\big)\frac{\partial(N_1(d_1)-1)}{\partial S} = \exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial S}$$

So

$$\Gamma_p = \Gamma_c = \frac{n(d_1)}{S\sigma\sqrt{T-t}}\exp\big(-q(T-t)\big) \tag{A.2.2}$$

So the value of gamma for both a put and a call is the same.

A.3 Delta

Delta is defined as the rate of change of option value with the underlying stock price.

For a European call we have:

$$\Delta_c = \frac{\partial c}{\partial S} = \frac{\partial}{\partial S}\big\{S\exp\big(-q(T-t)\big)N_1(d_1) - E\exp\big(-r(T-t)\big)N_1(d_2)\big\}$$

So

$$\Delta_c = \exp\big(-q(T-t)\big)\left\{N_1(d_1) + Sn(d_1)\frac{\partial d_1}{\partial S}\right\} - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial S} \tag{A.3.1}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial S}$ we obtain:

$$\Delta_c = \exp\big(-q(T-t)\big)N_1(d_1) \tag{A.3.2}$$

In similar manner we have for a European put:

$$\Delta_p = \frac{\partial p}{\partial S} = \frac{\partial}{\partial S}\big\{E\exp\big(-r(T-t)\big)\big(1-N_1(d_2)\big) - S\exp\big(-q(T-t)\big)\big(1-N_1(d_1)\big)\big\}$$

So

$$\Delta_p = -E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial S} - \exp\big(-q(T-t)\big)\left\{\big(1-N_1(d_1)\big) + Sn(d_1)\frac{\partial d_1}{\partial S}\right\} \tag{A.3.3}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial S}$ we obtain:

$$\Delta_p = \exp\big(-q(T-t)\big)\big\{N_1(d_1) - 1\big\} \tag{A.3.4}$$

A.4 Theta

Theta is defined as the rate of change of the option value with time.

For a European call option we have:

$$\begin{aligned}\Theta_c = \frac{\partial c}{\partial t} &= \frac{\partial}{\partial t}\big\{S\exp\big(-q(T-t)\big)N_1(d_1) - E\exp\big(-r(T-t)\big)N_1(d_2)\big\}\\ &= q\exp\big(-q(T-t)\big)SN_1(d_1) + \exp\big(-q(T-t)\big)Sn(d_1)\frac{\partial d_1}{\partial t}\\ &\quad - rE\exp\big(-r(T-t)\big)N_1(d_2) - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial t}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial t}$ we obtain:

$$\begin{aligned}\Theta_c &= q\exp\big(-q(T-t)\big)SN_1(d_1) - rE\exp\big(-r(T-t)\big)N_1(d_2)\\ &\quad + \exp\big(-q(T-t)\big)Sn(d_1)\frac{\partial d_1}{\partial t}\\ &\quad - E\exp\big(-r(T-t)\big)n(d_1)\frac{S}{E}\exp\big(r(T-t)\big)\\ &\quad \times \exp\big(-q(T-t)\big)\left\{\frac{\partial d_1}{\partial t} + \frac{\sigma}{2(T-t)}\right\}\\ &= q\exp\big(-q(T-t)\big)SN_1(d_1)\\ &\quad - rE\exp\big(-r(T-t)\big)N_1(d_2) - \frac{Sn(d_1)\sigma\exp(-q(T-t))}{2\sqrt{T-t}}\end{aligned}$$

Therefore the value of theta is:

$$\begin{aligned}\Theta_c &= \exp\big(-q(T-t)\big)\left\{q - SN_1(d_1)\frac{Sn(d_1)\sigma}{2\sqrt{T-t}}\right\}\\ &\quad - rE\exp\big(-r(T-t)\big)N_1(d_2)\end{aligned} \tag{A.4.1}$$

For a put we can similarly show that

$$\begin{aligned}\Theta_p = \frac{\partial p}{\partial t} &= \frac{\partial}{\partial t}\big\{E\exp\big(-r(T-t)\big)\big(1 - N_1(d_2)\big)\\ &\qquad - S\exp\big(-q(T-t)\big)\big(1 - N_1(d_1)\big)\big\}\\ \Theta_p &= rE\exp\big(-r(T-t)\big)\big(1 - N_1(d_2)\big) - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial t}\\ &\quad - qS\exp\big(-q(T-t)\big)\big(1 - N_1(d_1)\big) + S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial t}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial t}$ we obtain:

$$\begin{aligned}\Theta_p = {} & rE\exp\big(-r(T-t)\big)N_1(-d_2) - qS\exp\big(-q(T-t)\big)N_1(-d_1) \\ & - E\exp\big(-r(T-t)\big)\exp\big(r(T-t)\big) \\ & \times \exp\big(-q(T-t)\big)n(d_1)\frac{S}{E}\left\{\frac{\partial d_1}{\partial t} + \frac{\sigma}{2(T-t)}\right\} \\ & + S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial t}\end{aligned}$$

So we have:

$$\begin{aligned}\Theta_p = {} & -\exp\big(-q(T-t)\big)\left\{qSN_1(-d_1) + \frac{Sn(d_1)\sigma}{2\sqrt{T-t}}\right\} \\ & + rE\exp\big(-r(T-t)\big)N_1(-d_2)\end{aligned} \tag{A.4.2}$$

A.5 Rho

Rho is the rate of change of the option value with interest rate.

For a call we have:

$$\begin{aligned}\rho_c = \frac{\partial c}{\partial r} &= \frac{\partial}{\partial r}\big\{S\exp\big(-q(T-t)\big)N_1(d_1) - E\exp\big(-r(T-t)\big)N_1(d_2)\big\} \\ &= S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial r} + E(T-t)N_1(d_2) \\ &\quad - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial r}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial r}$ we obtain:

$$\rho_c = E(T-t)N_1(d_2) \tag{A.5.1}$$

For a European put we have:

$$\begin{aligned}\rho_p &= \frac{\partial p}{\partial r} \\ &= \frac{\partial}{\partial r}\big\{E\exp\big(-r(T-t)\big)\big(1-N_1(d_2)\big) - S\exp\big(-q(T-t)\big)\big(1-N_1(d_2)\big)\big\} \\ &= -E(T-t)\big(1-N_1(d_2)\big) - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial r} \\ &\quad + S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial r} \\ &= -E(T-t)N_1(-d_2) - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial r} \\ &\quad + S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial r}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial r}$ we obtain:

$$\rho_p = -E(T-t)N_1(-d_2) \tag{A.5.2}$$

A.6 Vega

Vega is the rate of change of option value with volatility. For a call we have:

$$\begin{aligned}\mathcal{V}_c &= \frac{\partial c}{\partial \sigma} \\ &= \frac{\partial}{\partial \sigma}\left\{S\exp\big(-q(T-t)\big)N_1(d_1) - E\exp\big(-r(T-t)\big)N_1(d_2)\right\} \\ &= S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial \sigma} - E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial r} \qquad \text{(A.6.1)}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial \sigma}$ we obtain:

$$\begin{aligned}\mathcal{V}_c = {} & S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial \sigma} \\ & - Sn(d_1)\exp\big(-q(T-t)\big)\left\{\frac{\partial d_1}{\partial \sigma} - \sqrt{T-t}\right\}\end{aligned}$$

Therefore

$$\mathcal{V}_c = S\exp\big(-q(T-t)\big)n(d_1)\sqrt{T-t} \qquad \text{(A.6.2)}$$

For a European put we have:

$$\begin{aligned}\mathcal{V}_p &= \frac{\partial c}{\partial \sigma} \\ &= \frac{\partial}{\partial \sigma}\left\{E\exp\big(-r(T-t)\big)\big(1-N_1(d_2)\big) - S\exp\big(-q(T-t)\big)\big(1-N_1(d_1)\big)\right\} \\ &= -E\exp\big(-r(T-t)\big)n(d_2)\frac{\partial d_2}{\partial \sigma} + S\exp\big(-q(T-t)\big)n(d_1)\frac{\partial d_1}{\partial \sigma}\end{aligned}$$

Substituting for $n(d_2)$ and $\frac{\partial d_2}{\partial \sigma}$ we obtain:

$$\mathcal{V}_p = S\exp\big(-q(T-t)\big)n(d_1)\sqrt{T-t} \qquad \text{(A.6.3)}$$

which is the same as for a call.

Appendix B: Barrier option integrals

B.1 The down and out call

We will now derive the formula for the value c_{do} of a European down and out call option with dividend yield q when the strike, E, satisfies $E > B$.

$$c_{\text{do}} = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \{S\exp(X) - E\} f(X > B)\,\mathrm{d}X \tag{B.1.1}$$

Substituting for $f(X > B)$ we have $c_{\text{do}} = I_A + I_B$ where:

$$I_A = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \{S\exp(X) - E\} \times \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\mathrm{d}X$$

and

$$I_B = -\frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \{S\exp(X) - E\} \times \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right)\mathrm{d}X$$

Now comparing I_A with Eq. (4.4.54) we can identify I_A as c, the price of a European call. That is:

$$I_A = S\exp(-q\tau)N_1(d_1) - E\exp(-r\tau)N_1(d_2) \tag{B.1.2}$$

where:

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_2 = \frac{\log(S/E) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

We now consider the term I_B, and let $I_B = I_C + I_D$ where:

$$I_C = -\frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}}\int_{X=\log(E/S)}^{\infty}\exp(X)\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\times\exp\left(\frac{2\log(B/S)(X-\log(B/S))}{\sigma^2\tau}\right)\mathrm{d}X$$

and

$$I_D = \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}}\int_{X=\log(E/S)}^{\infty}\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\times\exp\left(\frac{2\log(B/S)(X-\log(B/S))}{\sigma^2\tau}\right)\mathrm{d}X$$

We will first consider I_D and factor the integrand as follows:

$$\begin{aligned}
&-\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)\exp\left(\frac{2\log(B/S)(X-\log(B/S))}{\sigma^2\tau}\right)\\
&\quad=\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau\}^2-4\log(B/S)(X-\log(B/S))}{2\sigma^2\tau}\right)\\
&\quad=\exp\left(-\frac{\{X-(r-q-\sigma^2/2)\tau-2\log(B/S)\}^2}{2\sigma^2\tau}\right)\\
&\qquad\times\exp\left(\frac{4(r-q-\sigma^2/2)\tau\log(B/S)}{2\sigma^2\tau}\right)
\end{aligned}\tag{B.1.3}$$

This means that I_D can be expressed as:

$$\begin{aligned}
\therefore I_D &= \left(\frac{B}{S}\right)^{2((r-q)\sigma^2/2)/\sigma^2}\frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}}\\
&\quad\times\int_{X=\log(E/S)}^{\infty}\exp\left(-\frac{\{X-(r-q\sigma^2/2)\tau-2\log(B/S)\}^2}{2\sigma^2\tau}\right)\mathrm{d}X
\end{aligned}$$

Letting $u=(X-(r-q-\sigma^2/2)\tau-2\log(B/S))/(\sigma\sqrt{\tau})$ we have $\mathrm{d}X=\sigma\sqrt{\tau}\,\mathrm{d}u$ and

$$I_D = \left(\frac{B}{S}\right)^{2(r-q-\sigma^2/2)/\sigma^2}\frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}}\int_{u=k_3}^{\infty}\exp\left(-\frac{u^2}{2}\right)\mathrm{d}u$$

where

$$\begin{aligned}
k_3 &= \frac{\log(E/S)-(r-q-\sigma^2/2)\tau-2\log(B/S)}{\sigma\sqrt{\tau}}\\
&= \frac{\log(ES/B^2)-(r-q-\sigma^2/2)\tau}{\sigma\sqrt{\tau}}
\end{aligned}$$

So

$$I_D = \left(\frac{B}{S}\right)^{2r/\sigma^2-1}E\exp(-r\tau)N_1(-k_3)\tag{B.1.4}$$

Letting $d_3 = -k_3$ we have:

$$d_3 = \frac{\log(B^2/SE) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$I_D = \left(\frac{B}{S}\right)^{2r/\sigma^2 - 1} E\exp(-r\tau)N_1(d_3) \tag{B.1.5}$$

Now consider the term:

$$I_C = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\infty} \exp(X)\exp\left(-\frac{\{X - (r - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \mathrm{d}X$$

Now we have

$$\begin{aligned}
&\exp(X)\exp\left(-\frac{(X - (r - q - \sigma^2/2)\tau)^2}{2\sigma^2\tau}\right)\exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \\
&\quad = \exp\big(\big(-\big\{\big(X - \big(-qr - \sigma^2/2\big)\tau\big)^2 - 2\sigma^2\tau X \\
&\qquad\qquad - 4\log(B/S)X + 4\big(\log(B/S)\big)^2\big\}\big)/\big(2\sigma^2\tau\big)\big) \\
&\quad = \exp\big(\big(\big(\sigma^2\tau\big)^2 + 2\big(r - q - \sigma^2/2\big)\tau^2\sigma^2 \\
&\qquad\qquad + 4\big(r - q - \sigma^2/2\big)\tau\log(B/S) + 4\sigma^2\tau\log(B/S)\big)/\big(2\sigma^2\tau\big)\big) \\
&\qquad \times \exp\left(\frac{-\{X - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)\}^2}{2\sigma^2\tau}\right) \\
&\quad = \exp\big((r - q)\tau\big)\exp\left(\left\{\frac{2(r - q)}{\sigma^2} + 1\right\}\log\left(\frac{B}{S}\right)\right) \\
&\qquad \times \exp\left(\frac{-\{X - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)\}^2}{2\sigma^2\tau}\right) \\
&\quad = \exp\big((r - q)\tau\big)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} \\
&\qquad \times \exp\left(\frac{-\{X - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)\}^2}{2\sigma^2\tau}\right)
\end{aligned}$$

So we have:

$$\begin{aligned}
I_C = &-\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} \frac{S\exp(-q\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \\
&\times \int_{X=\log(E/S)}^{\infty} \exp\big(-\big\{X - \big(r - q - \sigma^2/2\big)\tau - \sigma^2\tau - 2\log(B/S)\big\}^2 \\
&\qquad \times \big(2\sigma^2\tau\big)^{-1}\big)\,\mathrm{d}X
\end{aligned}$$

Letting

$$u = \frac{X - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$

we have $\mathrm{d}X = \sigma\sqrt{\tau}\,\mathrm{d}u$ and

$$I_C = -S\exp(-q\tau)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} N_1(-k_4) \tag{B.1.6}$$

where

$$k_4 = \frac{\log(E/S) - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$

$$= \frac{\log(ES/B^2) - (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$\therefore I_C = -S\exp(-q\tau)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} N_1(-k_4)$$

or letting $d_4 = -k_4$ we have

$$d_4 = \frac{\log(B^2/ES) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$I_C = -S\exp(-q\tau)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} N_1(d_4) \tag{B.1.7}$$

Therefore the value for the down and out call option is:

$$c_{\mathrm{do}} = I_A + I_C + I_D = I_A - (-I_C - I_D)$$

Since $c_{\mathrm{do}} + c_{\mathrm{di}} = c$, where c is the value of vanilla call and c_{di} is the value of down and in call, we can write:

$$c_{\mathrm{do}} = c - c_{\mathrm{di}}$$

where

$$c_{\mathrm{di}} = S\exp(-q\tau)N_1(d_4)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1}$$

$$- E\exp(-r\tau)N_1(d_3)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2-1}$$

B.2 The up and out call

We will now derive the formula for a European up and out call option with dividend yield q when the strike, E, satisfies $B > E$.

$$c_{\mathrm{uo}} = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \left\{S\exp(X) - E\right\} f(X < B)\,\mathrm{d}X \tag{B.2.1}$$

Substituting for $f(X < B)$ we have $c_{\text{uo}} = I_A + I_B$ where:

$$I_A = \frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \left\{S\exp(X) - E\right\} \times \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \mathrm{d}X$$

and

$$I_B = -\frac{\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \times \int_{X=\log(E/S)}^{\log(B/S)} \left\{S\exp(X) - E\right\} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \mathrm{d}X$$

Letting $I_A = I_1 + I_2$ where

$$I_1 = \frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \exp(X) \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \mathrm{d}X$$

and

$$I_2 = \frac{-E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \mathrm{d}X$$

From our previous derivation of the Black–Scholes formula in Chapter 4 we have:

$$I_1 = \frac{S\exp(-q\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{u=k_1}^{k_2} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u = S\exp(-q\tau)\left\{N_1(k_2) - N_1(k_1)\right\}$$

where

$$k_1 = \frac{\log(E/S) - (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$k_2 = \frac{\log(B/S) - (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$I_2 = \frac{-E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{u=k_3}^{k_4} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u = -E\exp(-r\tau)\left\{N_1(k_4) - N_1(k_3)\right\}$$

where

$$k_3 = \frac{\log(E/S) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$k_4 = \frac{\log(B/S) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

Therefore,

$$I_A = S\exp(-q\tau)\big\{N_1(k_2) - N_1(k_1)\big\} - E\exp(-r\tau)\big\{N_1(k_4) - N_1(k_3)\big\}$$

Since $N_1(-x) = 1 - N_1(x)$ we have

$$N_1(k_2) - N_1(k_1) = N_1(-k_1) - N_1(-k_2)$$

so

$$\begin{aligned} I_A &= S\exp(-q\tau)\big\{N_1(d_1) - N_1(d_2)\big\} - E\exp(-r\tau)\big\{N_1(d_3) - N_1(d_4)\big\} \\ &= S\exp(-q\tau)N_1(d_1) - E\exp(-r\tau)N_1(d_3) \\ &\quad - S\exp(-q\tau)N_1(d_2) + E\exp(-r\tau)N_1(d_4) \end{aligned}$$

which gives:

$$I_A = c - S\exp(-q\tau)N_1(d_2) + E\exp(-r\tau)N_1(d_4) \tag{B.2.2}$$

where c is the value of a vanilla call and

$$d_1 = \frac{\log(S/E) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}, \qquad d_2 = \frac{\log(S/B) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$d_3 = \frac{\log(S/E) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}, \qquad d_4 = \frac{\log(S/B) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

Letting $I_B = I_C + I_D$ where:

$$I_C = -\frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \exp(X)\exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \mathrm{d}X$$

and

$$I_D = \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right) \times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \mathrm{d}X$$

In a similar manner to that in Section B.1 we have:

$$I_D = \left(\frac{B}{S}\right)^{2(r-q-\sigma^2/2)/\sigma^2} \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \times \int_{X=\log(E/S)}^{\log(B/S)} \exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau - 2\log(B/S)\}^2}{2\sigma^2\tau}\right) \mathrm{d}X$$

Letting

$$u = \frac{X - (r - q - \sigma^2/2)\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$

gives

$$I_D = \left(\frac{B}{S}\right)^{2(r-q-\sigma^2/2)/\sigma^2} \frac{E\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{u=k5}^{k_6} \exp\left(-\frac{u^2}{2}\right) \mathrm{d}u \tag{B.2.3}$$

where

$$k_5 = \frac{\log(E/S) - (r - q - \sigma^2/2)\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$
$$= \frac{\log(ES/B^2) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$k_6 = \frac{\log(B/S) - (r - q - \sigma^2/2)\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$
$$= \frac{\log(S/B) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and so

$$I_D = \left(\frac{B}{S}\right)^{2(r-q)/\sigma^2-1} E\exp(-r\tau)\big\{N_1(k_6) - N_1(k_5)\big\}$$

This can be re-expressed as:

$$I_D = \left(\frac{B}{S}\right)^{2(r-q)/\sigma^2-1} E\exp(-r\tau)\big\{N_1(d_5) - N_1(d_6)\big\} \tag{B.2.4}$$

where

$$d_5 = \frac{\log(B^2/ES) - (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$
$$d_6 = \frac{\log(B/S) + (r - q - \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

We now consider the term:

$$I_C = -\frac{S\exp(-r\tau)}{\sigma\sqrt{\tau}\sqrt{2\pi}} \int_{X=\log(E/S)}^{\log(B/S)} \exp(X)\exp\left(-\frac{\{X - (r - q - \sigma^2/2)\tau\}^2}{2\sigma^2\tau}\right)$$
$$\times \exp\left(\frac{2\log(B/S)(X - \log(B/S))}{\sigma^2\tau}\right) \mathrm{d}X$$

In a similar manner to Section B.1 we let

$$u = \frac{X - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$

which gives:

$$I_C = -S\exp(-r\tau)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} \big\{N_1(k_8) - N_1(k_7)\big\}$$

where

$$k_7 = \frac{\log(E/S) - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$
$$= \frac{\log(ES/B^2) - (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

and

$$k_8 = \frac{\log(B/S) - (r - q - \sigma^2/2)\tau - \sigma^2\tau - 2\log(B/S)}{\sigma\sqrt{\tau}}$$
$$= \frac{\log(S/B) - (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

This can be re-expressed as:

$$I_C = -S\exp(-r\tau)\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1}\{N_1(d_7) - N_1(d_8)\}$$

where

$$d_7 = \frac{\log(B^2/ES) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}},$$
$$d_8 = \frac{\log(B/S) + (r - q + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

So we have:

$$c_{\text{uo}} = I_A + I_C + I_D$$

which can be written as:

$$c_{\text{uo}} = c - c_{\text{ui}}$$

where c is the value of a vanilla call and c_{ui}, the value of an up and in call, is given by:

$$c_{\text{ui}} = S\exp(-q\tau)N_1(d_2) - E\exp(-r\tau)N_1(d_4)$$
$$- E\exp(-r\tau)\{N_1(d_5) - N_1(d_6)\}\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2-1}$$
$$+ S\exp(-r\tau)\{N_1(d_7) - N_1(d_8)\}\left(\frac{B}{S}\right)^{2(r-q)/\sigma^2+1} \quad \text{(B.2.5)}$$

Appendix C: Standard statistical results

C.1 The law of large numbers

Let $x_1, x_2, \ldots$ be a sequence of independent, identically distributed random variables (IID), each with expected value μ and variance σ^2. Define the sequence of averages

$$y_n = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n}, \quad n = 1, 2, \ldots$$

Then the law of large numbers states that y_n converges to μ as $n \to \infty$, that is $\mathrm{Var}[y_n] \to 0$.

The mean of y_n is:

$$E[y_n] = \frac{1}{n}\big(E[x_1] + E[x_2] + \cdots + E[x_n]\big) = \frac{1}{n} n\mu = \mu$$

For the variance of y_n we have:

$$\begin{aligned}\mathrm{Var}[y_n] &= \mathrm{Var}\left[\frac{\sum_{i=1}^{n} x_i}{n}\right] = \frac{1}{n^2}\,\mathrm{Var}\left[\sum_{i=1}^{n} x_i\right] \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}[x_i] = \frac{1}{n^2} n\sigma^2 = \frac{\sigma^2}{n}\end{aligned}$$

where we have used the fact that the variance of the sum of independent random variables is the sum of their variances; see Section C.2.

We have therefore shown that as $n \to \infty$, $\mathrm{Var}[y_n] \to 0$.

C.2 The central limit theorem

Let $x_1, x_2, \ldots$ be a sequence of independent, identically distributed random variables (IID), each with expected value μ and variance σ^2. If we define $u_i = x_i - \mu$ then:

$$E[u_i] = E[u] = 0, \qquad \mathrm{Var}[u_i] = E\big[u^2\big] = \sigma^2$$

Let

$$s_n = \sum_{i=1}^{n} u_i + n\mu$$

So

$$s_n = \sum_{i=1}^{n} x_i$$

We now introduce the normalized value z_n, as follows:

$$z_n = \frac{s_n - n\mu}{\sigma\sqrt{n}} = \frac{1}{\sigma\sqrt{n}} \sum_{i=1}^{n} u_i$$

The central limit theorem states that as n tends to infinity the probability distribution of z_n tends to a normal distribution with zero mean and unit variance; mathematically $z_n \to \mathrm{N}(0, 1)$ as $n \to \infty$.

Proof. From Section C.5, Eq. (C.5.3)

$$M_{z_n} = E\big[\exp(tz_n)\big] = E\left[\exp\left\{\frac{t}{\sigma\sqrt{n}} \sum_{i=1}^{n} u_i\right\}\right]$$

and using Eq. (C.5.5)

$$M_{z_n} = \left\{M_u\left(\frac{t}{\sigma\sqrt{n}}\right)\right\}^n$$

Equation (C.5.1) then yields:

$$M_u\left(\frac{t}{\sigma\sqrt{n}}\right) \approx 1 + \frac{t}{\sigma\sqrt{n}} E[u] + \frac{1}{2}\left(\frac{t}{\sigma\sqrt{n}}\right)^2 E\big[u^2\big] + \cdots$$

As $n \to \infty$, $\frac{t}{\sigma\sqrt{n}} \to 0$ and

$$M_u\left(\frac{t}{\sigma\sqrt{n}}\right) \to 1 + \frac{t}{\sigma\sqrt{n}} E[u] + \frac{1}{2}\left(\frac{t}{\sigma\sqrt{n}}\right)^2 E\big[u^2\big] = 1 + \frac{t^2}{2n}$$

Thus

$$\left\{M_u\left(\frac{t}{\sigma\sqrt{n}}\right)\right\}^n = \left(1 + \frac{t^2}{2n}\right)^n \to 1 + \frac{t^2}{2} \quad \text{as } n \to \infty$$

where we have used the fact that $t \ll 1$; see Grimmett and Welsh (1986).

We have therefore shown that as $n \to \infty$

$$M_{z_n}(t) \to 1 + \frac{t^2}{2} \to \mathrm{e}^{t^2/2}$$

However, from Section D.1 the moment generating function $M_z(t)$ for a standard normal distribution ($\mu = 0, \sigma^2 = 1$) is:

$$M_z(t) = \mathrm{e}^{t^2/2} \quad \text{where } z \sim \mathrm{N}(0, 1)$$

Thus we have proved that $z_n \to \mathrm{N}(0, 1)$ as $n \to \infty$. □

C.3 The variance and covariance of random variables

C.3.1 Variance

One variable

Let X be a variate from a given distribution, and Z be the following linear function of this variate:

$$Z = a + bX$$

where and a and b are constants. Then

$$E[Z] = E[a] + E[bX] = a + bE[X]$$

and

$$\begin{aligned}
\mathrm{Var}[Z] &= E\big[\big(Z - E[Z]\big)^2\big] \\
&= E\big[\big(a + bX - a - bE[X]\big)^2\big] \\
&= E\big[\big(bX - bE[X]\big)^2\big] \\
&= E\big[b^2\big(X - E[X]\big)^2\big] \\
&= b^2 E\big[\big(X - E[X]\big)^2\big]
\end{aligned}$$

Therefore the mean is $a + bE[X]$, and the variance is $b^2\,\mathrm{Var}[X]$.

Two variables

Let $Z = a + b_1X_1 + b_2X_2$, where a, b_1 and b_2 are constants.
Then the mean is $E[Z] = E[a]+E[b_1X_1]+E[b_2X_2] = a+b_1E[X_1]+b_2E[X_2]$.
The variance $\mathrm{Var}[Z]$ is computed as follows:

$$\begin{aligned}
\mathrm{Var}[Z] &= E\big[\big\{a + b_1X_1 + b_2X_2 - a - b_1E[X_1] - b_2E[X_2]\big\}^2\big] \\
&= E\big[\big\{b_1\big(X_1 - E[X_1]\big) + b_2\big(X_2 - E[X_2]\big)\big\}^2\big] \\
&= b_1^2 E\big[\big(X_1 - E[X_1]\big)^2\big] + b_2^2 E\big[\big(X_2 - E[X_2]\big)^2\big] \\
&\quad + 2b_1b_2 E\big[\big(X_1 - E[X_1]\big)\big]E\big[\big(X_2 - E[X_2]\big)\big] \\
&= b_1^2\,\mathrm{Var}[X_1] + b_2^2\,\mathrm{Var}[X_2] + 2b_1b_2\,\mathrm{Cov}[X_1, X_2]
\end{aligned}$$

where $\mathrm{Cov}[X_1, X_2]$ is the covariance between X_1 and X_2. If X_1 and X_2 are identical independently distributed random variables (IID) then $\mathrm{Cov}[X_1, X_2] = 0$, and we thus have:

$$\mathrm{Var}[Z] = b_1^2\,\mathrm{Var}[X_1] + b_2^2\,\mathrm{Var}[X_2]$$

Three variables

Let $Z = a + b_1X_1 + b_2X_2 = b_3X_3$, where a, b_1, b_2 and b_3 are constants.

Then the mean is $E[Z] = E[a]+E[b_1X_1]+E[b_2X_2]+E[b_3X_3] = a+b_1E[X_1]+b_2E[X_2]+b_3E[X_3]$.

The variance $\mathrm{Var}[Z]$ is computed as follows:

$$\begin{aligned}
\mathrm{Var}[Z] &= E\big[\{a + b_1X_1 + b_2X_2 + b_3X_3 - a \\
&\qquad - b_1E[X_1] - b_2E[X_2] - b_3E[X_3]\}^2\big] \\
&= E\big[\{b_1\big(X_1 - E[X_1]\big) + b_2\big(X_2 - E[X_2]\big) + b_3\big(X_3 - E[X_3]\big)\}^2\big] \\
&= b_1^2E\big[\big(X_1 - E[X_1]\big)^2\big] + b_2^2E\big[\big(X_2 - E[X_2]\big)^2\big] \\
&\quad + b_3^2E\big[\big(X_3 - E[X_3]\big)^2\big] \\
&\quad + 2b_1b_2E\big[\big(X_1 - E[X_1]\big)\big]E\big[\big(X_2 - E[X_2]\big)\big] \\
&\quad + 2b_2b_3E\big[\big(X_2 - E[X_2]\big)\big]E\big[\big(X_3 - E[X_3]\big)\big] \\
&\quad + 2b_1b_3E\big[\big(X_1 - E[X_1]\big)\big]E\big[\big(X_3 - E[X_3]\big)\big] \\
&= b_1^2\,\mathrm{Var}[X_1] + b_2^2\,\mathrm{Var}[X_2] + b_3^2\,\mathrm{Var}[X_2] + 2b_2b_3\,\mathrm{Cov}[X_2, X_3] \\
&\quad + 2b_1b_2\,\mathrm{Cov}[X_2, X_3] + 2b_1b_3\,\mathrm{Cov}[X_1, X_3]
\end{aligned}$$

If X_1, X_2 and X_3 are IID all the covariance terms are zero and the variance is:

$$\mathrm{Var}[Z] = b_1^2\,\mathrm{Var}[X_1] + b_2^2\,\mathrm{Var}[X_2] + b_3^2\,\mathrm{Var}[X_3]$$

Variance of n variables

We will now derive an expression for the sum of n IID random variables.

Let $Z = a + \sum_{i=1}^{n} b_iX_i$, where a and b_i, $i = 1, \ldots, n$, are constants.

Then we have: $E[Z] = E[a] + E[\sum_{i=1}^{n} b_iX_i] = a + \sum_{i=1}^{n} b_iE[X_i]$ and

$$\begin{aligned}
\mathrm{Var}[Z] &= E\left[\left\{a + \sum_{i=1}^{n} b_iX_i - a - \sum_{i=1}^{n} b_iE[X_i]\right\}^2\right] \\
&= E\left[\left\{\sum_{i=1}^{n} b_iX_i - \sum_{i=1}^{n} b_iE[X_i]\right\}^2\right] \\
&= E\left[\left\{\sum_{i=1}^{n} b_i\big(X_i - E[X_i]\big)\right\}^2\right] \\
&= \sum_{i=1}^{n} b_i^2E\big[\big(X_i - E[X_i]\big)^2\big] \\
&\quad + \sum_{i=1}^{n}\sum_{j=1(j\neq i)}^{n} b_ib_jE\big[\big(X_i - E[X_i]\big)\big(X_j - E[X_j]\big)\big]
\end{aligned}$$

$$= \sum_{i=1}^{n} b_i^2 \operatorname{Var}[X_i] + \sum_{i=1}^{n} \sum_{j=1(j\neq i)}^{n} b_i b_j \operatorname{Cov}[X_i, X_j]$$

As before if all the X variables are IID then the covariance terms are zero, and we have:

$$\operatorname{Var}[Z] = \sum_{i=1}^{n} b_i^2 \operatorname{Var}[x_i]$$

If in addition all the b_i terms are one and all the X variable have variance σ^2 we obtain:

$$\operatorname{Var}[Z] = \sum_{i=1}^{n} \operatorname{Var}[x_i] = n\sigma^2$$

C.3.2 Covariance

The covariance between two variables X and Y is defined by:

$$\begin{aligned}
\operatorname{Cov}[X, Y] &= E\big[\big(X - E[X]\big)\big(Y - E[Y]\big)\big] \\
&= E\big[XY - YE[X] - XE[Y] + E[X]E[Y]\big] \\
&= E[XY] - E[Y]E[X] - E[X]E[Y] + E[X]E[Y] \\
&= E[XY] - E[X]E[Y]
\end{aligned}$$

By symmetry it can be seen that $\operatorname{Cov}[X, Y] = \operatorname{Cov}[Y, X]$.

Two variables

Let $Z_1 = a + bX$ and $Z_2 = c + dY$, where a, b, c and d are constants.
We have:

$$\begin{aligned}
\operatorname{Cov}[Z_1, Z_2] &= \operatorname{Cov}[a + bX, c + dY] \\
&= E\big[(a + bX)(c + dY)\big] - E\big[(a + bX)\big]E\big[(c + dY)\big] \\
&= E[ac + bcX + adY + bdXY] - \big\{\big(a + bE[X]\big)\big(c + dE[Y]\big)\big\} \\
&= ac + bcE[X] + adE[Y] + bdE[XY] \\
&\quad - ac + bcE[X] - adE[Y] - bdE[X]E[Y] \\
&= bd\big\{E[XY] - E[X]E[Y]\big\}
\end{aligned}$$

$$\therefore \operatorname{Cov}[Z_1, Z_2] = bd \operatorname{Cov}[X, Y]$$

Three variables

Let $Z_1 = a + b_1X_1 + b_2X_2$ and $Z_2 = c + dY$, where a, b_1, b_2, c and d are constants.

We have:

$$\begin{aligned}
\text{Cov}[Z_1, Z_2] &= \text{Cov}[a + b_1X_1 + b_2X_2, c + dY] \\
&= E\big[(a + b_1X_1 + b_2X_2)(c + dY)\big] \\
&\quad - E\big[(a + b_1X_1 + b_2X_2)\big]E\big[(c + dY)\big] \\
&= E\big[(a + b_1X_1)(c + dY) + b_2X_2(c + dY)\big] \\
&\quad - \big\{E\big[(a + b_1X_1)\big]E[c + dY] + E[b_2X_2]E[c + dY]\big\} \\
&= E\big[(a + b_1X_1)(c + dY)\big] + E\big[b_2X_2(c + dY)\big] \\
&\quad - E\big[(a + b_1X_1)\big]E[c + dY] - E[b_2X_2]E[c + dY] \\
&= \big\{E\big[(a + b_1X_1)(c + dY)\big] - E\big[(a + b_1X_1)\big]E[c + dY]\big\} \\
&\quad - \big\{E\big[(b_2X_2)(c + dY)\big] - E[b_2X_2)]E[c + dY]\big\} \\
\therefore \text{Cov}[Z_1, Z_2] &= b_1d\,\text{Cov}[X_1, Y] + b_2d\,\text{Cov}[X_2, Y]
\end{aligned}$$

Four variables

Let $Z_1 = a + b_1X_1 + b_2X_2 + b_3X_3$ and $Z_2 = c + dY$, where a, b_1, b_2, b_3, c and d are constants.

We have:

$$\begin{aligned}
\text{Cov}[Z_1, Z_2] &= \text{Cov}[a + b_1X_1 + b_2X_2 + b_3X_3, c + dY] \\
&= E\big[(a + b_1X_1 + b_2X_2 + b_3X_3)(c + dY)\big] \\
&\quad - E\big[(a + b_1X_1 + b_2X_2 + b_3X_3)\big]E\big[(c + dY)\big] \\
&= E\big[(a + b_1X_1 + b_2X_2)(c + dY) + b_3X_3(c + dY)\big] \\
&\quad - \big\{E\big[(a + b_1X_1 + b_2X_2)\big]E[c + dY] + E[b_3X_3]E[c + dY]\big\} \\
&= E\big[(a + b_1X_1 + b_2X_2)(c + dY)\big] + E\big[b_3X_3(c + dY)\big] \\
&\quad - E\big[(a + b_1X_1 + b_2X_2)\big]E[c + dY] - E[b_3X_3]E[c + dY] \\
&= \big\{E\big[(a + b_1X_1 + b_2X_2)(c + dY)\big] \\
&\quad\quad - E\big[(a + b_1X_1 + b_2X_2)\big]E[c + dY]\big\} \\
&\quad\quad + \big\{E\big[(b_3X_3)(c + dY)\big] - E[b_3X_3]E[c + dY]\big\} \\
&= \text{Cov}\big[(a + b_1X_1 + b_2X_2), c + dY\big] + \text{Cov}[b_3X_3, c + dY] \\
\therefore \text{Cov}[Z_1, Z_2] &= b_1d\,\text{Cov}[X_1, Y] + b_2d\,\text{Cov}[X_2, Y] + b_3d\,\text{Cov}[X_3, Y]
\end{aligned}$$

Covariance of n variables

In a similar manner to that outlined above:

$$\text{Cov}\left[a + \sum_{i=1}^{n} b_iX_i, c + dY\right] = d\sum_{i=1}^{n} b_i\,\text{Cov}[X_i, Y]$$

For the most general case let: $Z_1 = a + \sum_{i=1}^{n} b_i X_i$ and $Z_2 = c + \sum_{j=1}^{M} b_j Y_j$.

So

$$\begin{aligned}\mathrm{Cov}[Z_1, Z_2] &= \mathrm{Cov}\left[a + \sum_{i=1}^{n} b_i X_i, c + \sum_{j=1}^{M} d_j Y_j\right] \\ &= \mathrm{Cov}\left[a + \sum_{i=1}^{n} b_i X_i, c + \sum_{j=1}^{M} d_j Y_j\right]\end{aligned}$$

So

$$\begin{aligned}\mathrm{Cov}\left[a + \sum_{i=1}^{n} b_i X_i, c + \sum_{j=1}^{M} d_j Y_j\right] &= \sum_{i=1}^{n} \mathrm{Cov}\left[b_i X_i, \sum_{j=1}^{M} d_j Y_j\right] \\ &= \sum_{i=1}^{n} b_i \,\mathrm{Cov}\left[X_i, \sum_{j=1}^{M} d_j Y_j\right] \\ &= \sum_{i=1}^{n} b_i \,\mathrm{Cov}\left[\sum_{j=1}^{M} d_j Y_j, X_i\right]\end{aligned}$$

$$\therefore \mathrm{Cov}[Z_1, Z_2] = \sum_{i=1}^{n}\left\{b_i \sum_{j=1}^{M} d_j \,\mathrm{Cov}[Y_j, X_i]\right\}$$

C.3.3 Covariance matrix

Let X denote the n element vector containing the random variates $X_i, i = 1, \ldots, n$. The mean and variance of the ith variate is then $E[X_i]$ and $E[(X_i - E[X_i])^2]$ respectively. The covariance $\mathrm{Cov}[X]_{ij}$ between the ith and jth variates is $E[(X_i - E[X_i])(X_j - E[X_j])]$. The elements of n by n covariance matrix $\mathrm{Cov}[X]$ are then:

$$\begin{aligned}&\mathrm{Cov}[X]_{ij} = E\big[\big(X_i - E[X_i]\big)\big(X_j - E[X_j]\big)\big], \\ &\quad i = 1, \ldots, n,\ j = 1, \ldots, n\end{aligned} \tag{C.3.1}$$

We will now show that $\mathrm{Cov}[X + A] = \mathrm{Cov}[X]$ where A is an n element vector containing the constants $A_i, i = 1, \ldots, n$. Since $E[X_i + A_i] = E[X_i] + A_i$ we obtain:

$$\begin{aligned}\mathrm{Var}\big[(X + A)_i\big] &= \mathrm{Var}[X_i + A_i] \\ &= E\big[\big(X_i + A_i - E[X_i + A_i]\big)^2\big] = E\big[\big(X_i - E[X_i]\big)^2\big]\end{aligned}$$

and

$$\begin{aligned}\mathrm{Cov}[X + A]_{ij} &= E\big[\big(X_i + A_i - E[X_i + A_i]\big)\big(X_j + A_j - E[X_j + A_j]\big)\big] \\ &= E\big[\big(X_i - E[X_i]\big)\big(X_j - E[X_j]\big)\big] \\ &= \mathrm{Cov}[X]_{ij}\end{aligned} \tag{C.3.2}$$

C.4 Conditional mean and covariance of normal distributions

Let $X = \left[\begin{smallmatrix} X_1 \\ X_2 \end{smallmatrix}\right]$ be distributed as $\mathrm{N}_p(\mu, \Sigma)$ with $\mu = \left[\begin{smallmatrix} \mu_1 \\ \mu_2 \end{smallmatrix}\right]$, and $\Sigma = \left[\begin{smallmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{smallmatrix}\right]$, and $|\Sigma_{22}| > 0$.

We will prove that the conditional distribution of X_1, given that $X_2 = x_2$, is normal and has:

Mean = $\mu_e s1 + \Sigma_{11}\Sigma_{22}^{-1}(x_2 - \mu_2)$, and covariance = $\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.

Let the inverse of Σ be Σ^{-1}, where:

$$\Sigma^{-1} = \begin{pmatrix} \Sigma^{11} & \Sigma^{12} \\ \Sigma^{21} & \Sigma^{22} \end{pmatrix} \tag{C.4.1}$$

So $\Sigma^{-1}\Sigma = I_p$, where I_p represents the $p \times p$ unit matrix, and:

$$\begin{pmatrix} \Sigma^{11} & \Sigma^{12} \\ \Sigma^{21} & \Sigma^{22} \end{pmatrix} \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \begin{pmatrix} I_q & 0 \\ 0 & I_{p-q} \end{pmatrix} \tag{C.4.2}$$

Multiplying out these matrices yields the following equations:

$$\Sigma^{11}\Sigma_{11} + \Sigma^{21}\Sigma_{21} = I_q \tag{C.4.3}$$

$$\Sigma^{21}\Sigma_{11} + \Sigma^{22}\Sigma_{22} = 0 \tag{C.4.4}$$

$$\Sigma^{11}\Sigma_{12} + \Sigma^{12}\Sigma_{22} = 0 \tag{C.4.5}$$

$$\Sigma^{21}\Sigma_{12} + \Sigma^{22}\Sigma_{22} = I_{p-q} \tag{C.4.6}$$

Multiplying Eq. (C.4.5) on the left by $(\Sigma^{11})^{-1}$ and on the right by Σ_{22}^{-1} gives:

$$\left(\Sigma^{11}\right)^{-1}\Sigma^{12} = -\Sigma_{12}\Sigma_{22}^{-1} \tag{C.4.7}$$

Multiplying Eq. (C.4.3) on the left by $(\Sigma^{11})^{-1}$ yields

$$\Sigma_{11} + \left(\Sigma^{11}\right)^{-1}\Sigma^{12}\Sigma_{21} = (\Sigma_{11})^{-1} \tag{C.4.8}$$

and substituting for $(\Sigma^{11})^{-1}\Sigma^{12}$ from Eq. (C.4.7) into Eq. (C.4.8) gives

$$(\Sigma_{11})^{-1} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{C.4.9}$$

The joint probability density function of x is:

$$f(x) = (2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right\}$$

writing x, μ and Σ^{-1} in their partitioned form and expanding gives:

$$\begin{aligned} f(x) = {} & (2\pi)^{-p/2}|\Sigma|^{-1/2} \\ & \times \exp\Bigg[-\frac{1}{2}\big\{(x_1-\mu_1)^\top\Sigma^{11}(x_1-\mu_1) + 2(x_1-\mu_1)^\top\Sigma^{12}(x_2-\mu_2) \\ & \quad + (x_2-\mu_2)^\top\Sigma^{22}(x_2-\mu_2)\big\}\Bigg] \end{aligned} \tag{C.4.10}$$

The conditional distribution of x_1 given the value of x_2 is thus obtained by dividing this density by the marginal density of x_2, and treating x_2 as a constant in the resulting expression. The only portion of the resultant that is not constant is the portion involving terms in x_1. It can easily be shown that:

$$f(x_1|x_2) \propto \exp\left[-\frac{1}{2}\left\{(x_1-\mu_1)^\top \Sigma^{11}(x_1-\mu_1) + 2(x_1-\mu_1)^\top \Sigma^{12}(x_2-\mu_2)\right\}\right]$$

where the constant of proportionality is obtained using $\int f(x_1|x_2)\,\mathrm{d}x_1 = 1$.

If we let $\mathcal{G} = (x_1-\mu_1)^\top \Sigma^{11}(x_1-\mu_1) + 2(x_1-\mu_1)^\top \Sigma^{12}(x_2-\mu_2)$ we then obtain:

$$\begin{aligned}
\mathcal{G} &= (x_1-\mu_1)^\top \Sigma^{11}(x_1-\mu_1) + (x_1-\mu_1)^\top \Sigma^{12}(x_2-\mu_2) \\
&\quad + (x_2-\mu_2)^\top \Sigma^{21}(x_1-\mu_1) \\
\mathcal{G} &= \left\{x_1-\mu_1+\left(\Sigma^{11}\right)^{-1}\Sigma^{12}(x_2-\mu_2)\right\}^\top \\
&\quad \times \Sigma^{11}\left\{x_1-\mu_1+\left(\Sigma^{11}\right)^{-1}\Sigma^{12}(x_2-\mu_2)\right\} \\
&\quad - (x_2-\mu_2)^\top \Sigma^{21}\left(\Sigma^{12}\right)^{-1}(x_2-\mu_2)
\end{aligned} \tag{C.4.11}$$

where, for instance, we have used the fact that the scalar quantity

$$\left\{(x_1-\mu_1)^\top \Sigma^{12}(x_2-\mu_2)\right\} = (x_2-\mu_2)^\top \Sigma^{21}(x_1-\mu_1)$$

Since the last term in Eq. (C.4.11) only involves constants (as far as $f(x_1|x_2)$ is concerned), it follows that:

$$f(x_1|x_2) \propto \exp\left[-\frac{1}{2}\left\{x_1-\mu_1+\left(\Sigma^{11}\right)^{-1}\Sigma^{12}(x_2-\mu_2)\right\}^\top \times \Sigma^{11}\left\{x_1-\mu_1+\left(\Sigma^{11}\right)^{-1}\Sigma^{12}(x_2-\mu_2)\right\}\right]$$

which is the density of a multivariate normal distribution that has a mean of $\mu_1 - (\Sigma^{11})^{-1}\Sigma^{12}(x_2-\mu_2)$, which from Eq. (C.4.7) can be expressed as $\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2)$. The covariance matrix is $(\Sigma^{11})^{-1}$, which from Eq. (C.4.9) can be written as $\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.

C.5 Moment generating functions

If x is a random variable with probability distribution $f_x(x)$ then the moment generating function $M_x(t)$ is defined by:

$$M_x(t) = E\left[\mathrm{e}^{tx}\right] = \int_{-\infty}^{\infty} \mathrm{e}^{tx} f_x(x)\,\mathrm{d}x$$

We can expand the above expression as follows:

$$E\left[\mathrm{e}^{tx}\right] = E\left[1 + tx + \frac{1}{2}(tx)^2 + \cdots\right]$$

$$M_x(t) = 1 + tE[x] + \frac{1}{2}t^2E\left[x^2\right] + \cdots \tag{C.5.1}$$

Now

$$\frac{\mathrm{d}^k(M_x(t))}{\mathrm{d}t^k} = \frac{\mathrm{d}^k}{\mathrm{d}t^k}\left\{E\left[\mathrm{e}^{tx}\right]\right\} = E\left[\frac{\mathrm{d}^k\mathrm{e}^{tx}}{\mathrm{d}t^k}\right] = E\left[x^k\mathrm{e}^{tx}\right]$$

For $t = 0$ we thus have:

$$\left.\frac{\mathrm{d}^k(M_x(t))}{\mathrm{d}t^k}\right|_{t=0} = \frac{\mathrm{d}^k(M_x(0))}{\mathrm{d}t^k} = E\left[x^k\mathrm{e}^0\right] = E\left[x^k\right] \tag{C.5.2}$$

Moment generating function of a linear function of a random variable x

If the random variable y is defined as: $y = ax + b$ then the moment generating function of y, $M_y(t)$ is obtained as follows:

$$M_y(t) = M_{ax+b}(t) = E\left[\mathrm{e}^{ty}\right] = E\left[\mathrm{e}^{atx+bt}\right] = \mathrm{e}^{bt}E\left[\mathrm{e}^{tx}\right]$$

Therefore:

$$M_y(t) = \mathrm{e}^{bt}M_x(at) \tag{C.5.3}$$

Moment generating function of a linear combination of random variables

Let $z = x + y$ where x and y are independent random variables. Then

$$M_z(t) = E\left[\mathrm{e}^{tz}\right] = E\left[\mathrm{e}^{x+y}\right] = E\left[\mathrm{e}^{tx}\mathrm{e}^{ty}\right]$$

Since x and y are independent:

$$E\left[\mathrm{e}^{tx}\mathrm{e}^{ty}\right] = E\left[\mathrm{e}^{tx}\right]E\left[\mathrm{e}^{ty}\right] = M_x(t)M_y(t)$$

More generally if $s_n = \sum_{i=1}^n x_i$ where $x_i, i = 1, \ldots, n$, are independent variables then:

$$M_{s_n}(t) = \prod_{i=1}^{n} M_{x_i}(t) \tag{C.5.4}$$

If $x_i, i = 1, \ldots, n$, are IID then we have

$$M_{s_n}(t) = E\left[\exp\left(t\sum_{i=1}^{n} x_i\right)\right] = \left(E\left[\mathrm{e}^{tx}\right]\right)^n = \left(M_x(t)\right)^n \tag{C.5.5}$$

Appendix D: Statistical distribution functions

D.1 The normal (Gaussian) distribution

Here we describe some properties of the normal distribution. If x comes from a normal distribution, then the associated moment generating function, $M_x(t)$, is given by:

$$\begin{aligned} M_x(t) &= E\left[\mathrm{e}^{tx}\right] \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp(tx)\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\mathrm{d}x \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp\left(-\frac{(x-\mu)^2-2\sigma^2tx}{2\sigma^2}\right)\mathrm{d}x \end{aligned}$$

Now completing the square we have:

$$\begin{aligned} -\frac{1}{2\sigma^2}\left\{(x-\mu)^2-2t\sigma^2x\right\} &= -\frac{1}{2\sigma^2}\left\{x^2+\mu^2-2\mu x-2t\sigma^2\right\} \\ &= -\frac{1}{2\sigma^2}\left\{\left(x-\sigma^2t-\mu\right)^2-2\mu t\sigma^2-\sigma^4t^2\right\} \\ &= \mu t+\frac{\sigma^2t^2}{2}-\frac{1}{2\sigma^2}\left\{\left(x-\sigma^2t-\mu\right)^2\right\} \end{aligned}$$

We thus have:

$$E\left[\mathrm{e}^{tx}\right] = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\mu t+\frac{\sigma^2t^2}{2}\right)\int_{x=-\infty}^{x=\infty}\exp\left(-\frac{(x-\sigma^2t-\mu)^2}{2\sigma^2}\right)\mathrm{d}x$$

Now letting $y = x-\sigma^2t-\mu$, $\mathrm{d}x = \mathrm{d}y$ and

$$\begin{aligned} E\left[\mathrm{e}^{tx}\right] &= \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\mu t+\frac{\sigma^2t^2}{2}\right)\int_{y=-\infty}^{y=\infty}\exp\left(-\frac{y^2}{2\sigma^2}\right)\mathrm{d}y \\ &= \frac{1}{\sigma\sqrt{2\pi}}\sigma\sqrt{2\pi}\exp\left(\mu t+\frac{\sigma^2t^2}{2}\right) \\ &= \exp\left(\mu t+\frac{\sigma^2t^2}{2}\right) \end{aligned}$$

where we have used (see Section E.1) the fact that

$$\int_{-\infty}^{\infty} \exp(-ay^2)\,\mathrm{d}y = \sqrt{\frac{\pi}{a}}$$

Thus the moment generating function $M_x(t)$ for a normal distribution with mean μ and variance σ^2 is:

$$M_x(t) = \exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)$$

D.1.1 Some elementary results involving the mean and variance of a normal distribution

From first principles we have:

- The mean:

$$E[x] = \frac{1}{\sigma\sqrt{2\pi}} \int_{x=-\infty}^{x=\infty} x \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x$$

  Letting $y = x - \mu$ we have $\mathrm{d}x = \mathrm{d}y$ and $x = y + \mu$; therefore:

$$E[x] = \frac{1}{\sigma\sqrt{2\pi}} \int_{y=-\infty}^{y=\infty} (y+\mu) \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y$$

$$E[x] = \mu \frac{1}{\sigma\sqrt{2\pi}} \int_{y=-\infty}^{y=\infty} \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y + \frac{1}{\sigma\sqrt{2\pi}} \int_{y=-\infty}^{y=\infty} y \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y$$

  Since

$$\int_{-\infty}^{\infty} y \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y = 0$$

  we have using the integral result (i) in Section E.1 with $a = 1/(2\sigma^2)$:

$$E[x] = \mu \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y = \mu$$

- The variance:

$$E\left[x^2\right] = \frac{1}{\sigma\sqrt{2\pi}} \int_{x=-\infty}^{x=\infty} x^2 \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x$$

  Letting $y = x - \mu$ we have $\mathrm{d}x = \mathrm{d}y$ and $x^2 = y^2 + 2\mu y + \mu^2$; therefore:

$$E\left[x^2\right] = \frac{1}{\sigma\sqrt{2\pi}} \int_{y=-\infty}^{y=\infty} \left(y^2 + 2\mu y + \mu^2\right) \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y$$

$$E[x^2] = \mu^2 \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y + \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y\infty} y^2 \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y$$

So

$$E[x^2] = \mu^2 + \sigma^2$$

where we have used (see Section E.1 result (ii) with $a = 1/(2\sigma^2)$) that

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} y^2 \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y = \sigma^2$$

Therefore:

$$\mathrm{Var}[x] = E[x^2] - \left(E[x]\right)^2 = \mu^2 + \sigma^2 - \mu^2 = \sigma^2$$

The mean and variance can also be obtained by using the moment generating function, $M_x(t)$.

From Section C.5:

$$E[x] = \left.\frac{\mathrm{d}M_x(t)}{\mathrm{d}t}\right|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t}\left\{\exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)\right\}_{t=0}$$
$$= \left(\mu + \sigma^2 t\right) \exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)\Big|_{t=0} = \mu$$

also

$$E[x^2] = \left.\frac{\mathrm{d}^2 M_x(t)}{\mathrm{d}t^2}\right|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t}\left\{\left(\mu + \sigma^2 t\right) \exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)\right\}_{t=0}$$
$$E[x^2] = \left\{\exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)\left(\mu + \mu\sigma^2 t\right)^2 + \sigma^2\right\}_{t=0}$$
$$= \mu^2 + \sigma^2$$

These results are the same as those we previously derived from first principles:

$$E[x] = \mu \quad \text{and} \quad E[x^2] = \mu^2 + \sigma^2$$

D.2 The lognormal distribution

If the variable x follows a lognormal distribution then the probability density function $f(x)$ is given by:

$$\frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right) \tag{D.2.1}$$

where $x > 0$. Here we denote the lognormal distribution for x as: $x \sim \Lambda(\mu, \sigma^2)$. Setting $y = \log(x)$ it can be seen that $y \sim \mathrm{N}(\mu, \sigma^2)$. Thus if x is a lognormal

distribution $\Lambda(\mu, \sigma^2)$ then $\log(x)$ is a normal distribution with mean μ and variance σ^2. Conversely if $y \sim \mathrm{N}(\mu, \sigma^2)$ then the distribution for $x = \mathrm{e}^y$ is $x \sim \Lambda(\mu, \sigma^2)$.

The expectation of the tth moment (where t is a positive integer) of x is thus:

$$E\left[x^t\right] = \frac{1}{\sigma\sqrt{2\pi}} \int_{x=-\infty}^{x=\infty} x^t \frac{1}{x} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right) \mathrm{d}x$$

Using $y = \log(x)$ we have:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}\log(x)}{\mathrm{d}x} = \frac{1}{x}, \quad \mathrm{d}x = x\,\mathrm{d}y, \quad \text{and} \quad x^t = \left(\mathrm{e}^y\right)^t = \mathrm{e}^{ty}$$

Thus,

$$E\left[x^t\right] = E\left[\mathrm{e}^{ty}\right] = M_y(t) = \frac{1}{\sigma\sqrt{2\pi}} \int_{y=-\infty}^{y=\infty} \mathrm{e}^{ty} \exp\left(-\frac{(y - \mu)^2}{2\sigma^2}\right) \mathrm{d}y$$

where $M_y(t)$ is the moment generating function of a normal distribution with mean μ and variance σ^2.

From Section D.1:

$$E\left[x^t\right] = M_y(t) = \exp\left(\mu t + \frac{\sigma^2 t^2}{2}\right)$$

Therefore if $x \sim \Lambda(\mu, \sigma^2)$ then:

For $t = 1$

$$E[x] = \exp\left(\mu + \frac{\sigma^2}{2}\right) \tag{D.2.2}$$

for $t = 2$

$$E\left[x^2\right] = \exp\left(2\mu + 2\sigma^2\right) \tag{D.2.3}$$

and the variance is obtained using

$$\mathrm{Var}[x] = E\left[x^2\right] - \left(E[x]\right)^2 = \exp\left(2\mu + 2\sigma^2\right) - \exp\left(2\mu + \sigma^2\right)$$

So

$$\mathrm{Var}[x] = \exp\left(2\mu + \sigma^2\right)\left(\exp\left(\sigma^2\right) - 1\right)$$

or

$$\mathrm{Var}[x] = \left(E[x]\right)^2\left(\exp\left(\sigma^2\right) - 1\right) \tag{D.2.4}$$

NOTE. If $x_1 = \exp(\mu + \sigma Z)$, where $Z \sim \mathrm{N}(0, 1)$ then $x_1 \sim \Lambda(\mu, \sigma^2)$. So $E[x_1]$ and $\mathrm{Var}[x_1]$ are given by Eqs. (D.2.2) and (D.2.4).

D.3 The Student's t distribution

This section derives an expression for the kurtosis of the Student's t distribution.

Since the Student's t distribution density function is:

$$f(\epsilon_i) = \mathcal{K}\left[1 + \frac{{\epsilon_i}^2}{h_i(\nu-2)}\right]^{-(\nu+1)/2}$$

where

$$\mathcal{K} = \frac{\Gamma((\nu+1)/2)(\nu-2)^{-1/2}h_i^{-1/2}}{\pi^{1/2}\Gamma(\nu/2)}$$

we have:

$$\begin{aligned} E\left[\epsilon_i^2\right] &= 2\mathcal{K}\int_0^\infty \frac{\epsilon_i^2\,\mathrm{d}\epsilon_i}{\left(1+\epsilon_i^2/(h_i(\nu-2))\right)^{(\nu+1)/2}} \\ &= 2\mathcal{K}\big(h_i(\nu-2)\big)^{(\nu+1)/2}\int_0^\infty \frac{\epsilon_i^2\,\mathrm{d}\epsilon_i}{(h_i(\nu-2)+\epsilon_i^2)^{(\nu+1)/2}} \end{aligned}$$

Using the standard integrals in Section E.1 with $a = 2$, $b = 2$, $c = (\nu+1)/2$ and $m = (\nu-2)h_i$ gives:

$$\frac{m^{(a+1-bc)/b}}{b} = \frac{(h_i(\nu-2))^{(2-\nu)/2}}{2}, \qquad \Gamma\left(\frac{a+1}{b}\right) = \Gamma\left(\frac{3}{2}\right),$$
$$\Gamma\left(c - \frac{a+1}{b}\right) = \Gamma\left(\frac{\nu-2}{2}\right), \qquad \Gamma(c) = \Gamma\left(\frac{\nu+1}{2}\right)$$

This gives

$$E\left[\epsilon_i^2\right] = 2\mathcal{K}\big(h_i(\nu-2)\big)^{(\nu+1)/2}\left\{\frac{(h_i(\nu-2))^{(2-\nu)/2}\sqrt{\pi}\Gamma((\nu-2)/2)}{4\Gamma((\nu+1)/2)}\right\}$$

Substituting for $\mathcal{K}$ and simplifying we obtain:

$$E\left[\epsilon_i^2\right] = \frac{h_i(\nu-2)\Gamma((\nu-1)/2)}{\Gamma(\nu/2)}$$

But

$$\left(\frac{\nu-2}{2}\right)\Gamma\left(\frac{\nu-2}{2}\right) = \Gamma\left(\frac{\nu-1}{2}+1\right) = \Gamma\left(\frac{\nu}{2}\right)$$

So

$$E\left[\epsilon_i^2\right] = \frac{h_i(\nu-2)\Gamma(\nu/2)}{2(\nu-2)\Gamma(\nu/2)} = h_i$$

Similarly we have:

$$E\left[\epsilon_i^4\right] = 2\mathcal{K}\int_0^\infty \frac{\epsilon_i^4\,\mathrm{d}\epsilon_i}{\left(1+\epsilon_i^2/(h_i(\nu-2))\right)^{(\nu+1)/2}}$$
$$= 2\mathcal{K}\left(h_i(\nu-2)\right)^{(\nu+1)/2}\int_0^\infty \frac{\epsilon_i^4\,\mathrm{d}\epsilon_i}{(h_i(\nu-2)+\epsilon_i^2)^{(\nu+1)/2}}$$

Using the standard integrals in Section E.1 with $a=4$, $b=2$, $c=(\nu+1)/2$, and $m=(\nu-2)h_i$ gives:

$$\frac{m^{(a+1-bc)/b}}{b} = \frac{(h_i(\nu-2))^{(4-\nu)/2}}{2}, \qquad \Gamma\left(\frac{a+1}{b}\right) = \Gamma\left(\frac{5}{2}\right),$$
$$\Gamma\left(c-\frac{a+1}{b}\right) = \Gamma\left(\frac{\nu-4}{2}\right), \qquad \Gamma(c) = \Gamma\left(\frac{\nu+1}{2}\right)$$

and

$$E\left[\epsilon_i^4\right] = 2\mathcal{K}\left(h_i(\nu-2)\right)^{(\nu+1)/2}\left\{\frac{(h_i(\nu-2))^{(4-\nu)/2}3\sqrt{\pi}\Gamma((\nu-4)/2)}{8\Gamma((\nu+1)/2)}\right\}$$

Substituting for $\mathcal{K}$ and simplifying we obtain:

$$E\left[\epsilon_i^4\right] = \frac{3h_i(\nu-2)^2\Gamma((\nu-4)/2)h_i^2}{4\Gamma(\nu/2)}$$

But

$$\left(\frac{\nu-4}{2}\right)\Gamma\left(\frac{\nu-4}{2}\right) = \Gamma\left(\frac{\nu-2}{2}\right)$$

and

$$\left(\frac{\nu-2}{2}\right)\Gamma\left(\frac{\nu-2}{2}\right) = \Gamma\left(\frac{\nu}{2}\right)$$

Therefore:

$$\Gamma\left(\frac{\nu-4}{2}\right) = \frac{4\Gamma(\nu/2)}{(\nu-4)(\nu-2)}$$

So

$$E\left[\epsilon_i^4\right] = \frac{3(\nu-2)^2 4\Gamma(\nu/2)h_i^2}{4\Gamma(\nu/2)(\nu-4)(\nu-2)} = \frac{3(\nu-2)h_i^2}{\nu-4}$$

The kurtosis is then:

$$\aleph = \frac{E[e_i^4]}{(E[e_i^2])^2} = \frac{3(\nu-2)h_i^2}{(\nu-4)h_i^2} = \frac{3(\nu-2)}{\nu-4} \tag{D.3.1}$$

D.4 The general error distribution

This section proves various relations for the generalized error distribution.

The density function for the generalized error distribution is:

$$f(\epsilon_i) = \mathcal{K} \exp\left(-\frac{1}{2}\left|\frac{\epsilon_i}{\lambda}\right|^a\right) \quad \text{where } \mathcal{K} = \frac{a}{\lambda 2^{(1+1/a)}\Gamma(1/a)} \tag{D.4.1}$$

D.4.1 Value of λ for variance h_i

Calculation of the scale factor λ required for a generalized error distribution with mean zero and variance h_i.

The variance of the distribution, $E(\epsilon_i^2)$, is given by:

$$E(\epsilon_i^2) = \mathcal{K}\int_{-\infty}^{\infty} \epsilon_i^2 \exp\left(-\frac{1}{2}\left|\frac{\epsilon_i}{\lambda}\right|^a\right) \mathrm{d}\epsilon_i = 2\mathcal{K}\int_0^{\infty} \epsilon_i^2 \exp\left(-\frac{1}{2}\left(\frac{\epsilon_i}{\lambda}\right)^a\right) \mathrm{d}\epsilon_i$$

Using the standard integrals in Section E.1 with $n = 2$, $p = a$, and $b = \frac{1}{2}(\frac{1}{\lambda})^a$ gives:

$$h_i = \frac{2\mathcal{K}}{a}\Gamma\left(\frac{3}{a}\right)\left\{\frac{1}{2}\left(\frac{1}{\lambda}\right)^a\right\}^{-3/a}$$

which after some simplification yields:

$$h_i = \frac{2\mathcal{K}2^{3/a}\lambda^3}{a}\Gamma\left(\frac{3}{a}\right)$$

Substituting for $\mathcal{K}$ and simplifying then gives:

$$h_i = \lambda^2 2^{2/a}\frac{\Gamma(3/a)}{\Gamma(1/a)}$$

The required value of λ is therefore:

$$\lambda = \left\{h_i 2^{-2/a}\frac{\Gamma(1/a)}{\Gamma(3/a)}\right\}^{1/2}$$

D.4.2 The kurtosis

$$E(\epsilon_i^4) = \mathcal{K}\int_{-\infty}^{\infty} \epsilon_i^4 \exp\left(-\frac{1}{2}\left|\frac{\epsilon_i}{\lambda}\right|^a\right) \mathrm{d}\epsilon_i = 2\mathcal{K}\int_0^{\infty} \epsilon_i^4 \exp\left(-\frac{1}{2}\left(\frac{\epsilon_i}{\lambda}\right)^a\right) \mathrm{d}\epsilon_i$$

However, from standard mathematical tables:

$$\int_0^{\infty} \epsilon_i^4 \exp\left(-b\epsilon_i^p\right) = \frac{\Gamma(k)}{pb^k}$$

where $p = a$, $b = \frac{1}{2}(\frac{1}{\lambda})^a$, and $k = 5/a$ which gives:

$$E\left[\epsilon_i^4\right] = \frac{2\mathcal{K}2^{5/a}\lambda^5}{a}\Gamma\left(\frac{5}{a}\right) = 2^{2/a}\lambda^2 h_i \frac{\Gamma(5/a)}{\Gamma(3/a)}$$

From Section E.1 we have:

$$E\left[\epsilon_i^2\right] = h_i = \frac{2\mathcal{K}2^{3/a}\lambda^3}{a}\Gamma\left(\frac{3}{a}\right) \quad \text{and} \quad \lambda^2 = \frac{h_i 2^{-2/a}\Gamma(1/a)}{\Gamma(3/a)}$$

Therefore:

$$E\left[\epsilon_i^4\right] = h_i^2\frac{\Gamma(5/a)\Gamma(1/a)}{\Gamma(3/a)\Gamma(3/a)}$$

which gives the kurtosis as:

$$\aleph = \frac{E[\epsilon_i^4]}{(E[\epsilon_i^2])^2} = \frac{h_i^2}{h_i^2}\frac{\Gamma(5/a)\Gamma(1/a)}{\Gamma(3/a)\Gamma(3/a)} = \frac{\Gamma(5/a)\Gamma(1/a)}{\Gamma(3/a)\Gamma(3/a)}$$

D.4.3 The distribution for shape parameter, a

If the distribution has variance h_i then, from Section D.4.1:

$$\lambda = \left(\frac{2^{-2/a}\Gamma(1/a)h_i}{\Gamma(3/a)}\right)^{1/2}$$

Now for $0 < x < 1$ we have $\Gamma(1+x) = 1 + a_1x + a_2x^2 + a_3x^3 + \cdots$, where the coefficients are $|a_i| < 1$ (see Abramowitz and Stegun (1968)).

Since $x\Gamma(x) = \Gamma(1+x)$, to third order in x, we have:

$$x\Gamma(x) = 1 + a_1x + a_2x^2 + a_3x^3$$

This gives $\Gamma(x) = \frac{1}{x} + a_1 + a_2x + a_3x^2$, and $\Gamma(x) \approx \frac{1}{x}$ as $x \to 0$.

So as $a \to \infty$ we have the following:

$$2^{(1+1/a)} \approx 2, \qquad 2^{-2/a} \approx 1, \qquad \frac{1}{\Gamma(1/a)} \approx \frac{1}{a},$$
$$\frac{\Gamma(1/a)}{\Gamma(3/a)} \approx \frac{3a}{a} = 3 \quad \text{and} \quad \frac{\Gamma(5/a)}{\Gamma(3/a)} \approx \frac{3a}{5a} = \frac{3}{5}$$

The kurtosis is then:

$$\aleph = \frac{\Gamma(5/a)\Gamma(1/a)}{\Gamma(3/a)\Gamma(3/a)} = \frac{9}{5}$$

Also as $a \to \infty$, $\lambda \approx (3h_i)^{1/2}$, and for the range $-(3h_i)^{1/2} < \epsilon_i < (3h_i)^{1/2}$, we have:

$$\left|\frac{\epsilon_i}{\lambda}\right|^a \approx \left|\frac{\epsilon_i}{(3h_i)^{1/2}}\right| \approx 0 \quad \text{and therefore} \quad \exp\left(-\frac{1}{2}\left|\frac{\epsilon_i}{\lambda}\right|^a\right) \approx 1$$

Substituting the above results into Eq. (D.4.1), the probability density function reduces to:

$$f(\epsilon_i) \approx \frac{1}{2(3h_i)^{1/2}}$$

which is a uniform distribution $\mathrm{U}(-(3h_i)^{1/2}, (3h_i)^{1/2})$, with lower limit $-(3h_i)^{1/2}$ and upper limit $-(3h_i)^{1/2}$.

Appendix E: Mathematical reference

E.1 Standard integrals

$$\int_0^\infty \exp\left(-ay^2\right) \mathrm{d}y = \frac{1}{2}\sqrt{\frac{\pi}{a}} \tag{i}$$

$$\int_0^\infty y \exp\left(-ay^2\right) \mathrm{d}y = \frac{1}{2} \tag{ii}$$

$$\int_0^\infty y^2 \exp\left(-ay^2\right) \mathrm{d}y = \frac{1}{4a}\sqrt{\frac{\pi}{a}} \tag{iii}$$

$$\int_0^\infty y^4 \exp\left(-ay^2\right) \mathrm{d}y = \frac{3}{8a^2}\sqrt{\frac{\pi}{a}} \tag{iv}$$

$$\int_0^\infty y^{2n} \exp\left(-ay^2\right) \mathrm{d}y = \frac{1 \times 3 \times 5 \times \cdots \times (2n-1)}{2^{n+1}a^n}\sqrt{\frac{\pi}{a}} \tag{v}$$

$$\int_0^\infty \epsilon_i^n \exp\left(-b\epsilon_i^p\right) = \frac{\Gamma(k)}{pb^k}, \quad \text{where } n > -1,\ p > 0, b > 0 \text{ and } k = \frac{n+1}{p} \tag{vi}$$

$$\int_0^\infty \frac{\epsilon_i^a d\epsilon_i}{(m+\epsilon_i^b)^c} = \frac{m^{(a+1-bc)/b}}{b} \frac{\Gamma((a+1)/b)\Gamma(c-(a+1)/b)}{\Gamma(c)} \quad \text{where } a > -1, b > 0, m > 0 \text{ and } c > \frac{a+1}{b} \tag{vii}$$

E.2 Gamma function

For more detail see Abramowitz and Stegun (1968).

$$\Gamma(1+x) = x!$$

$$x\Gamma(x) = \Gamma(x+1)$$

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

$$\Gamma\left(\frac{3}{2}\right) = \frac{\sqrt{\pi}}{2}$$
$$\Gamma\left(\frac{5}{2}\right) = \frac{3\sqrt{\pi}}{4}$$
$$\frac{\partial \log(\Gamma(x))}{\partial x} = \psi(x)$$

For $0 \leqslant x \leqslant 1$ we have:

$$\Gamma(1+x) = 1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

where $a_1 = -0.5748$, $a_2 = 0.9512$, $a_3 = -0.6998$, $a_4 = 0.4245$ and $a_5 = -0.1010$.

E.3 The cumulative normal distribution function

In this section we show that the cumulative normal distribution function, $N_1(x)$, is related to the complementary error function, $\text{erfc}(x)$, by the following equation:

$$N_1(x) = \frac{1}{2}\,\text{erfc}\left(-\frac{x}{\sqrt{2}}\right) \tag{E.3.1}$$

If we let the error function be represented by $\text{erf}(x)$ then we have:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^{\infty} \exp\left(-t^2\right) \mathrm{d}t$$

Now we have the following:

$$\text{erfc}(x) = 1 - \text{erf}(x), \qquad \text{erf}(-x) = -\,\text{erf}(x),$$
$$\text{erf}(\infty) = 1 \quad \text{and} \quad \text{erfc}(-x) = 2 - \text{erfc}(x)$$

We will consider the integral

$$I(x) = \frac{2}{\sqrt{\pi}} \int_{-\infty}^{x} \exp\left(-t^2\right) \mathrm{d}t$$
$$= \frac{2}{\sqrt{\pi}} \int_{-\infty}^{0} \exp\left(-t^2\right) \mathrm{d}t + \frac{2}{\sqrt{\pi}} \int_0^{x} \exp\left(-t^2\right) \mathrm{d}t$$

Since

$$\frac{2}{\sqrt{\pi}} \int_{-\infty}^{0} \exp\left(-t^2\right) \mathrm{d}t = 1$$

we therefore have

$$I(x) = 1 + \text{erf}(x) = 1 + \left\{1 - \text{erfc}(x)\right\} = 2 - \text{erfc}(x)$$

Substituting for $\operatorname{erfc}(x)$ we obtain:

$$I(x) = 2 - \{2 - \operatorname{erfc}(-x)\} = \operatorname{erfc}(-x)$$

So we have

$$\operatorname{erfc}(-x) = \frac{2}{\sqrt{\pi}} \int_{-\infty}^{x} \exp(-t^2)\, \mathrm{d}t \tag{E.3.2}$$

Now the cumulative normal distribution is defined as

$$N_1(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-t^2)\, \mathrm{d}t$$

Letting $u = t\sqrt{2}$, we have $\mathrm{d}u = \sqrt{2}\,\mathrm{d}t$, and for the upper limit we have $x = t\sqrt{2}$, or $t = x/\sqrt{2}$.

This integral becomes:

$$N_1(x) = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{t=x/\sqrt{2}} \exp(-t^2)\sqrt{2}\, \mathrm{d}t \tag{E.3.3}$$

So from Eq. (E.3.2) we have:

$$N_1(x) = \frac{1}{2} \operatorname{erfc}\left(-\frac{x}{\sqrt{2}}\right)$$

We also note that:

$$N_1(-x) = 1 - N_1(x)$$

E.4 Arithmetic and geometric progressions

Arithmetic progression

The sum of the first n terms of an arithmetic progression is:

$$s_n = \frac{n}{2}\{2a_1 + (n-1)d\} \tag{E.4.1}$$

where a_1 is the first term, and d is the common difference; that is, the terms in the sequence are: $a_1, a_1 + d, a_1 + 2d, a_1 + 3d, \ldots$.

Geometric progression

The sum of the first n terms of a geometric progression is:

$$s_n = \frac{a_1(1 - r^n)}{1 - r} \tag{E.4.2}$$

where a_1 is the first term, and r is the common ratio; that is, the terms in sequence are: $a_1, a_1 r, a_1 r^2, a_1 r^3, \ldots$.

This page intentionally left blank

Appendix F:

Black–Scholes finite-difference schemes

F.1 The general case

In this section we consider the stability of the finite-difference schemes described in Chapter 5. It is assumed that the grid contains n_s asset points, and we will denote the time dependent option values at the ith and $(i+1)$th time instants by the $n_s - 2$ element vectors X^i and X^{i+1} respectively. We can therefore write:

$$T_1 X^i = T_2 X^{i+1} \tag{F.1.1}$$

where T_1 and T_2 are $(n_s - 2) \times (n_s - 2)$ tridiagonal matrices, and x_k^i, $k = 1, \ldots, n_s - 2$, will be used to denote the elements of the vector X^i.

The option values at the ith time instant are computed from those at the $(i+1)$th time instant by using

$$X^i = T_1^{-1} T_2 X^{i+1} \tag{F.1.2}$$

However, Eq. (F.1.2) is only *stable* if the eigenvalues of the $(n_s - 2) \times (n_s - 2)$ matrix $T_1^{-1} T_2$ all have modulus less than one (see Smith (1985)).

F.2 The log transformation and a uniform grid

We will now prove that the *implicit* finite difference method, $\Theta_m = 0$, when used on the log transformed Black–Scholes equation with a uniform grid is *unconditionally stable* which means that the stability does not depend on the values of σ, Δt, ΔZ, etc.

From Chapter 5 the finite-difference scheme is described by the following tridiagonal system:

$$\begin{pmatrix} B & C & 0 & 0 & 0 & 0 \\ A & B & C & 0 & 0 & 0 \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & A & B & C \\ 0 & 0 & 0 & 0 & A & B \end{pmatrix} \begin{pmatrix} x_1^i \\ x_2^i \\ . \\ . \\ x_{s-1}^i \\ x_{s-2}^i \end{pmatrix}$$

$$= \begin{pmatrix} \overline{B} & \overline{C} & 0 & 0 & 0 & 0 \\ \overline{A} & \overline{B} & \overline{C} & 0 & 0 & 0 \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & \overline{A} & \overline{B} & \overline{C} \\ 0 & 0 & 0 & 0 & \overline{A} & \overline{B} \end{pmatrix} \begin{pmatrix} x_1^{i+1} \\ x_2^{i+1} \\ . \\ . \\ x_{s-3}^{i+1} \\ x_{s-2}^{i+1} \end{pmatrix}$$

where

$$A = \frac{(1-\Theta_m)\Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\} \tag{F.2.1}$$

$$B = 1 + (1-\Theta_m)\Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\} \tag{F.2.2}$$

$$C = -\frac{(1-\Theta_m)\Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\} \tag{F.2.3}$$

$$\overline{A} = -\frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\} \tag{F.2.4}$$

$$\overline{B} = 1 - \Theta_m \Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\} \tag{F.2.5}$$

$$\overline{C} = \frac{\Theta_m \Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\} \tag{F.2.6}$$

As in Chapter 5, $b = r - q - \dfrac{\sigma^2}{2}$ and $r > 0$.

Substituting $\Theta_m = 0$ into Eqs. (F.2.1)–(F.2.6) we have $\overline{A} = \overline{C} = 0$, $\overline{B} = 1$, and

$$A = \frac{\Delta t}{2\Delta Z^2}\left\{b\Delta Z - \sigma^2\right\}, \qquad B = 1 + \Delta t\left\{r + \frac{\sigma^2}{\Delta Z^2}\right\},$$
$$C = -\frac{\Delta t}{2\Delta Z^2}\left\{b\Delta Z + \sigma^2\right\}$$

The finite-difference scheme is thus represented by the equations

$$\begin{pmatrix} B & C & 0 & 0 & 0 & 0 \\ A & B & C & 0 & 0 & 0 \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & A & B & C \\ 0 & 0 & 0 & 0 & A & B \end{pmatrix} \begin{pmatrix} x_1^i \\ x_2^i \\ . \\ . \\ x_{s-1}^i \\ x_{s-2}^i \end{pmatrix} = \begin{pmatrix} 1 & & 0 & 0 & 0 & 0 \\ 0 & 1 & & 0 & 0 & 0 \\ 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1^{i+1} \\ x_2^{i+1} \\ . \\ . \\ x_{s-3}^{i+1} \\ x_{s-2}^{i+1} \end{pmatrix}$$

or in matrix notation

$$X^i = {T_1}^{-1} X^{i+1} \tag{F.2.7}$$

where $T_2 = I$ in Eq. (F.1.1).

As mentioned in Section F.1, Eq. (F.2.7) is stable if the modulus of all the eigenvalues of T_1^{-1} are less than one. We will now show that this is in fact the case.

If the eigenvalues of T_1 are $\lambda_k, k = 1, \ldots, n_s - 2$, then the eigenvalues of T_1^{-1} are $\lambda_k^{-1}, k = 1, \ldots, n_s - 2$. This means that the system is stable if all the eigenvalues of T_1 have a modulus *greater* than one. This result can be proved by considering the eigenvalue with the smallest modulus, $\lambda_{\min}$. If $|\lambda_{\min}| > 1$ then the result is proved.

Now the eigenvalues of T_1, see Smith (1985), are given by:

$$\lambda_k = 1 + \Delta t\left(r + \frac{\sigma^2}{\Delta Z^2}\right) + 2\sqrt{AC}\cos\left(\frac{k\pi}{n_s - 2 + 1}\right) \qquad k = 1, \ldots, n_s - 2, \tag{F.2.8}$$

where the term

$$2\sqrt{AC} = \sqrt{\frac{\Delta t^2(\sigma^4 - b^2\Delta Z^2)}{\Delta Z^4}} \tag{F.2.9}$$

It can be seen that if $b^2\Delta Z^2 > \sigma^4$ then the eigenvalues are complex and if $\sigma^4 \geqslant b^2\Delta Z^2$ then eigenvalues are real. We will consider each of these cases in turn.

Complex eigenvalues: $b^2\Delta Z^2 > \sigma^4$

We will represent the kth complex eigenvalue as:

$$\lambda_k = R + \mathrm{i}Y$$

where the real part is

$$R = 1 + \Delta t\left(r + \frac{\sigma^2}{\Delta Z^2}\right)$$

and the imaginary part is

$$Y = 2\sqrt{AC}\cos\left(\frac{k\pi}{n_s - 2 + 1}\right)$$

Since

$$|\lambda_k| > |R| + |Y| \quad \text{and} \quad |R| > 1,$$

we conclude that

$$|\lambda_{\min}| > 1$$

Real eigenvalues: $\sigma^4 \geqslant b^2\Delta Z^2$

In this case the kth eigenvalue is real, and from Eq. (F.2.8) we have:

$$\lambda_k > 1 + \Delta t\left(r + \frac{\sigma^2}{\Delta Z^2}\right) - 2\sqrt{AC}$$

Since $b^2\Delta^2 > 0$ from Eq. (F.2.9) we have

$$2\sqrt{AC} < \sqrt{\frac{\sigma^4\Delta t^2}{\Delta Z^4}}$$

or

$$\left|2\sqrt{AC}\right| < \frac{\sigma^2\Delta t}{\Delta Z^2}$$

So

$$\lambda_{\min} > 1 + \Delta t\left(r + \frac{\sigma^2}{\Delta Z^2}\right) - \frac{\sigma^2\Delta t}{\Delta Z^2}$$

Therefore we have

$$|\lambda_{\min}| > 1 + r\Delta t$$

and, since $r > 0$, we have:

$$|\lambda_{\min}| > 1$$

Appendix G:

The Brownian bridge: alternative derivation

Here we provide an alternative derivation of the Brownian bridge equation given in Chapter 2.

Let a Brownian process have values W_{t_0} at time t_0 and W_{t_1} at time t_1. We want to find the conditional distribution of W_t, where $t_0 < t < t_1$. This distribution will be denoted by $P(W_t|\{W_{t_0}, W_{t_1}\})$, to indicate that W_t is conditional on the end values W_{t_0} and W_{t_1}.

We have:

$$P(W_t|W_{t_0}) = \frac{1}{\sqrt{2\pi(t-t_0)}} \exp\left\{-\frac{(W_t - W_{t_0})^2}{2(t-t_0)}\right\}$$

The joint distribution of W_t and W_{t_1} given W_{t_0} is:

$$\begin{aligned}
&P\left(\{W_t, W_{t_1}\}|W_{t_0}\right) \\
&\quad = P(W_{t_1}|W_t)P(W_t|W_{t_0}) \\
&\quad = \frac{1}{\sqrt{2\pi(t-t_0)(t_1-t)}} \exp\left\{-\frac{(W_t - W_{t_0})^2}{2(t-t_0)} - \frac{(W_{t_1} - W_t)^2}{2(t_1-t)}\right\} \\
&\quad = \frac{1}{\sqrt{2\pi(t-t_0)(t_1-t)}} \exp\left\{-\frac{1}{2}\left(\frac{(W_t - W_{t_0})^2}{2(t-t_0)} + \frac{(W_{t_1} - W_t)^2}{2(t_1-t)}\right)\right\}
\end{aligned}$$

Similarly

$$P(W_{t_1}|W_{t_0}) = \frac{1}{\sqrt{2\pi(t_1-t_0)}} \exp\left\{-\frac{(W_{t_1} - W_{t_0})^2}{2(t_1-t_0)}\right\}$$

Now we have:

$$\begin{aligned}
&P\left(W_t|\{W_{t_0}, W_{t_1}\}\right) \\
&\quad = \frac{P(\{W_t, W_{t_1}\}|W_{t_0})}{P(W_{t_1}|W_{t_0})} \\
&\quad = \frac{1}{\sqrt{2\pi}}\sqrt{\frac{t_1-t_0}{(t-t_0)(t_1-t)}} \\
&\qquad \times \exp\left\{-\frac{1}{2}\left(\frac{(W_t - W_{t_0})^2}{2(t-t_0)} + \frac{(W_{t_1} - W_t)^2}{2(t_1-t)} - \frac{(W_{t_1} - W_{t_0})^2}{2(t_1-t_0)}\right)\right\}
\end{aligned}$$

For ease of reference we will write the above equation as:

$$P\big(W_t|\{W_{t_0}, W_{t_1}\}\big) = \frac{1}{\sqrt{2\pi}}\sqrt{\frac{t_1 - t_0}{(t - t_0)(t_1 - t)}}\exp\{A\}$$

We now consider the terms in the exponent A.

$$A = -\frac{1}{2}\Bigg(\frac{(X - X_{t_0})^2(t_1 - t)(t_1 - t_0)}{(t - t_0)(t_1 - t)(t_1 - t_0)} + \frac{(X_1 - X)^2(t - t_0)(t_1 - t_0)}{(t - t_0)(t_1 - t)(t_1 - t_0)} - \frac{(X_1 - X_{t_0})^2(t - t_0)(t_1 - t)}{(t - t_0)(t_1 - t)(t_1 - t_0)}\Bigg\}$$

Dividing top and bottom of the above expression for A by $(t_1 - t_0)^2$ we then obtain:

$$A = -\frac{1}{2V}\Big\{\big(W_t^2 + W_{t_0}^2 - 2W_tW_{t_0}\big)\frac{t_1 - t}{t_1 - t_0} + \big(W_{t_1}^2 + W_t^2 - 2W_tW_{t_1}\big)\frac{t - t_0}{t_1 - t_0} - \big(W_{t_1}^2 + W_{t_0}^2 - 2W_{t_1}W_{t_0}\big)\frac{(t - t_0)(t_1 - t)}{(t_1 - t_0)^2}\Big\}$$

where

$$V = \frac{(t - t_0)(t_1 - t)}{t_1 - t_0}$$

So

$$A = -\frac{1}{2V}\Bigg(W_t^2\Big\{\frac{t_1 - t}{t_1 - t_0} + \frac{t - t_0}{t_1 - t_0}\Big\} + W_{t_1}^2\Big\{\frac{t - t_0}{t_1 - t_0} - \frac{(t - t_0)(t_1 - t)}{(t_1 - t_0)^2}\Big\} + W_{t_0}^2\Big\{\frac{t - t_0}{t_1 - t_0} - \frac{(t - t_0)(t_1 - t)}{(t_1 - t_0)^2}\Big\} - 2W_tW_{t_0}\Big\{\frac{t_1 - t}{t_1 - t_0}\Big\} - 2W_tW_{t_1}\Big\{\frac{t - t_0}{t_1 - t_0}\Big\} + 2W_{t_1}W_{t_0}\Big\{\frac{(t - t_0)(t_1 - t)}{(t_1 - t_0)^2}\Big\}\Bigg)$$

We now show that A can be expressed as quadratic form:

$$B = -\frac{1}{2V}(W_t - \mu)^2 = -\frac{1}{2V}\big(W_t^2 + \mu^2 - 2\mu W_t\big)$$

where

$$V = \frac{(t - t_0)(t_1 - t)}{t_1 - t_0} \quad \text{and} \quad \mu = W_{t_0}\frac{t_1 - t}{t_1 - t_0} + W_{t_1}\frac{t - t_0}{t_1 - t_0}$$

Therefore we have:

$$B = -\frac{1}{2V}\Bigg(W_t^2 + \Big\{W_{t_0}\frac{t_1 - t}{t_1 - t_0} + W_{t_1}\frac{t - t_0}{t_1 - t_0}\Big\}^2 - 2W_t\Big\{W_{t_0}\frac{t_1 - t}{t_1 - t_0} + W_{t_1}\frac{t - t_0}{t_1 - t_0}\Big\}\Bigg)$$

Expanding and gathering terms we obtain:

$$B = -\frac{1}{2V}\left(W_t^2 + W_{t_0}^2 \frac{(t_1 - t)^2}{(t_1 - t_0)^2} + W_{t_1}^2 \frac{(t - t_0)^2}{(t_1 - t_0)^2} + 2W_{t_0} W_{t_1} \frac{(t_1 - t)(t - t_0)}{(t_1 - t_0)^2} - 2W_t W_{t_0} \frac{t_1 - t}{t_1 - t_0} - 2W_t W_{t_1} \frac{t - t_0}{t_1 - t_0} \right)$$

Comparing coefficients of A and B we have:

Coefficients for W_t^2

$$A: \; -\frac{1}{2V}\left\{ \frac{t_1 - t}{t_1 - t_0} + \frac{t - t_0}{t_1 - t_0} \right\} = -\frac{1}{2V}\left\{ \frac{t_1 - t + t - t_0}{t_1 - t_0} \right\} = -\frac{1}{2V}$$

$$B: \; -\frac{1}{2V}$$

Coefficients for $W_{t_0}^2$

$$A: \; -\frac{1}{2V}\left\{ \frac{t_1 - t}{t_1 - t_0} - \frac{(t - t_0)(t_1 - t)}{(t_1 - t_0)^2} \right\}$$

$$= -\frac{1}{2V}\left\{ \frac{(t_1 - t)(t_1 - t_0) - (t - t_0)(t_1 - t)}{(t_1 - t_0)^2} \right\}$$

$$A: \; -\frac{1}{2V}\left\{ \frac{(t_1 - t)(t_1 - t_0 - t + t_0)}{(t_1 - t_0)^2} \right\} = -\frac{1}{2V}\left\{ \frac{(t_1 - t)^2}{(t_1 - t_0)^2} \right\}$$

$$B: \; -\frac{1}{2V}\left\{ \frac{(t_1 - t)^2}{(t_1 - t_0)^2} \right\}$$

Coefficients for $W_{t_1}^2$

$$A: \; -\frac{1}{2V}\left\{ \frac{(t - t_0)(t_1 - t_0) - (t - t_0)(t_1 - t)}{(t_1 - t_0)^2} \right\}$$

$$= -\frac{1}{2V}\left\{ \frac{(t - t_0)(t_1 - t_0 - t_1 + t)}{(t_1 - t_0)^2} \right\}$$

$$A: \; -\frac{1}{2V}\left\{ \frac{(t_1 - t)^2}{(t_1 - t_0)^2} \right\}$$

$$B: \; -\frac{1}{2V}\left\{ \frac{(t - t_0)^2}{(t_1 - t_0)^2} \right\}$$

The remaining coefficients in A and B for $W_{t_0} W_{t_1}$, $W_t W_{t_1}$ and $W_{t_0} W_t$ are identical.

We have thus shown that:

$$P\left(W_t|\{W_{t_0}, W_{t_1}\}\right) = \frac{1}{\sqrt{2\pi}}\sqrt{\frac{t_1 - t_0}{(t - t_0)(t_1 - t)}}\exp\left\{-\frac{(W_t - \mu)^2}{2V}\right\}$$
$$= \frac{1}{\sqrt{2\pi V}}\exp\left\{-\frac{(W_t - \mu)^2}{2V}\right\}$$

Thus the conditional distribution of W_t is a Gaussian with mean

$$\mu = W_{t_0}\frac{t_1 - t}{t_1 - t_0} + W_{t_1}\frac{t - t_0}{t_1 - t_0}$$

and variance

$$V = \frac{(t - t_0)(t_1 - t)}{t_1 - t_0}$$

and we can obtain a variate $\widehat{W}_t$ from this distribution by using:

$$\widehat{W}_t = W_{t_0}\frac{t_1 - t}{t_1 - t_0} + W_{t_1}\frac{t - t_0}{t_1 - t_0} + \sqrt{\frac{(t - t_0)(t_1 - t)}{t_1 - t_0}}Z, \quad \text{where } Z \sim \mathrm{N}(0, 1)$$

Appendix H:
Brownian motion: more results

H.1 Some results concerning Brownian motion

Here we will prove some facts concerning Brownian motion.

If the Brownian motion has zero drift then:

$$\mathrm{d}X_t = \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_t, \quad \mathrm{d}Z_t \sim \mathrm{N}(0,1) \tag{H.1.1}$$

and

$$P\left(m_t^X \leqslant b, X_t \geqslant x\right) = N_1\left(\frac{2b-x}{\sigma\sqrt{t}}\right) \tag{H.1.2}$$

where m_t^X denotes the minimum value of X_t over the time interval $[0, t]$, $b \leqslant 0$, and $x \geqslant 0$.

When the Brownian motion has nonzero drift

$$\mathrm{d}\overline{X}_t = \nu\,\mathrm{d}t + \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_t, \quad \mathrm{d}Z_t \sim \mathrm{N}(0,1) \tag{H.1.3}$$

and the following equations are satisfied by $m_t^{\overline{X}}$ and $\overline{X}_t$:

$$P(\overline{X}_t \leqslant K) = N_1\left(\frac{K-\nu t}{\sigma\sqrt{t}}\right) \tag{H.1.4}$$

$$P(\overline{X}_t \geqslant K) = N_1\left(\frac{\nu t-K}{\sigma\sqrt{t}}\right) \tag{H.1.5}$$

$$P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\right) = \exp\left(\frac{2\nu b}{\sigma^2}\right)N_1\left(\frac{2b-x+\nu t}{\sigma\sqrt{t}}\right) \tag{H.1.6}$$

$$P\left(m_t^{\overline{X}} \geqslant b, \overline{X}_t \geqslant x\right) = N_1\left(\frac{\nu t-x}{\sigma\sqrt{t}}\right) - \exp\left(\frac{2\nu b}{\sigma^2}\right)N_1\left(\frac{2b-x+\nu t}{\sigma\sqrt{t}}\right) \tag{H.1.7}$$

$$P\left(m_t^{\overline{X}} \leqslant b\right) = N_1\left(\frac{b-\nu t}{\sigma\sqrt{t}}\right) + \exp\left(\frac{2\nu b}{\sigma^2}\right)N_1\left(\frac{b+\nu t}{\sigma\sqrt{t}}\right) \tag{H.1.8}$$

$$P\left(m_t^{\overline{X}} \geqslant b\right) = N_1\left(\frac{\nu t-b}{\sigma\sqrt{t}}\right) - \exp\left(\frac{2\nu b}{\sigma^2}\right)N_1\left(\frac{b+\nu t}{\sigma\sqrt{t}}\right) \tag{H.1.9}$$

where K is a constant, $\overline{X} \geqslant 0$, $b \leqslant 0$, and $P(\mathit{condition})$ denotes the probability associated with the appropriate *condition*, i.e., $\overline{X}_t \geqslant K$, $m_t^{\overline{X}} \geqslant b$, $\overline{X}_t \geqslant x$, etc.

The conditional probability density function associated with $P(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x)$ is

$$p\left(\left\{m_{t_1,t_2}^{\overline{X}} \leqslant b, \overline{X}_{t_2}\right\} | \overline{X}_{t_1}\right) = \frac{1}{\sigma\sqrt{2\pi\Delta t}} \exp\left(\frac{2\nu(b - \overline{X}_{t_1})}{\sigma^2}\right) \exp\left(-\frac{(\overline{X}_{t_1} + \overline{X}_{t_2} - 2b + \nu\Delta t)^2}{2\sigma^2\Delta t}\right) \tag{H.1.10}$$

where $t_2 \geqslant t_1$ and $\Delta t = t_2 - t_1$.

H.2 Proof of Eq. (H.1.2)

From Eq. (H.1.1):

$$X_t = \sigma\sqrt{t} \quad Z_t, Z_t \sim \mathrm{N}(0, 1)$$

where $X_0 = 0$. We will derive the probability of events $m_t^X \leqslant b$ and $X_t \geqslant x$ occurring. For event m_t^X to occur there must be a time τ at which $X_\tau \leqslant b$, where $0 < \tau \leqslant t$. At time τ, instead of continuing with the original Brownian motion, X_t we will consider the *reflected* motion X_t^R defined by:

$$X_s^R = X_s, \qquad s < \tau$$
$$X_s^R = 2b - X_s, \quad s \geqslant \tau$$

Therefore, before time τ the motion X_s is identical to X_s^R. For $s \geqslant \tau$ the *coordinates* of X_s^R are obtained by reflecting those of X_s about the level b. The event $X_t \geqslant x$ is thus equivalent to the event $X_t^R \leqslant 2b - x$ (remember $b \leqslant 0$ and $x \geqslant 0$). However, the event $X_t^R \leqslant 2b - x$ only occurs if $m_t^X \leqslant b$ also occurs, giving:

$$P\left(X_t^R \leqslant 2b - x\right) = P\left(m_t^X \leqslant b, X_t \leqslant x\right)$$

At time τ we have

$$X_\tau^R = 2b - X_\tau \tag{H.2.1}$$

and after time τ

$$X_{\tau+\gamma}^R = 2b - X_{\tau+\gamma}, \quad \gamma > 0 \tag{H.2.2}$$

Thus subtracting Eq. (H.2.1) from Eq. (H.2.2) gives:

$$X_{\tau+\gamma}^R - X_\tau^R = X_\tau - X_{\tau+\gamma}$$
$$X_{\tau+\gamma}^R - X_\tau^R = -(X_{\tau+\gamma} - X_\tau) \tag{H.2.3}$$

However, we know that:

$$(X_{\tau+\gamma} - X_\tau) \sim \mathrm{N}\left(0, \sigma^2\gamma\right) \tag{H.2.4}$$

So

$$\left(X_{\tau+\gamma}^R - X_\tau^R\right) \sim -\mathrm{N}\left(0, \sigma^2\gamma\right)$$

which means that:

$$\left(X^R_{\tau+\gamma} - X^R_\tau\right) \sim \mathrm{N}\left(0, \sigma^2\gamma\right) \tag{H.2.5}$$

Since the left-hand sides of Eqs. (H.2.4) and (H.2.5) have the same distribution, and X_t satisfies the three Brownian properties given in Section 2.1, we can write:

$$P\left(X^R_t \leqslant 2b - x\right) = P(X_t \leqslant 2b - x) = N_1\left(\frac{2b - x}{\sigma\sqrt{t}}\right)$$

Therefore:

$$P\left(m^X_t \leqslant b, X_t \geqslant x\right) = N_1\left(\frac{2b - x}{\sigma\sqrt{t}}\right)$$

H.3 Proof of Eq. (H.1.4)

From Eq. (H.1.3) $\overline{X}_t = \nu t + \sigma\sqrt{t}Z_t$, $Z_t \sim \mathrm{N}(0, 1)$.

So we can write:

$$\begin{aligned} P(\overline{X}_t \leqslant K) &= P\left(\nu t + \sigma\sqrt{t}Z_t \leqslant K\right) \\ &= P\left(Z_t \leqslant \frac{K - \nu t}{\sigma\sqrt{t}}\right) \\ &= N_1\left(\frac{K - \nu t}{\sigma\sqrt{t}}\right), \quad Z_t \sim \mathrm{N}(0, 1) \end{aligned}$$

H.4 Proof of Eq. (H.1.5)

We know that $P(\overline{X}_t \geqslant K) = 1 - P(\overline{X}_t \leqslant K)$.

Substituting from Eq. (H.1.4) gives:

$$P(\overline{X}_t \geqslant K) = 1 - N_1\left(\frac{K - \nu t}{\sigma\sqrt{t}}\right)$$

Since $1 - N_1(x) = N_1(-x)$ we obtain:

$$P(\overline{X}_t \geqslant K) = N_1\left(\frac{\nu t - K}{\sigma\sqrt{t}}\right)$$

H.5 Proof of Eq. (H.1.6)

From Eq. (H.1.1):

$$\mathrm{d}X_t = \sigma\sqrt{\mathrm{d}t}\,\mathrm{d}Z_t, \quad \mathrm{d}Z_t \sim \mathrm{N}(0, 1)$$

This can be expressed as zero drift Brownian motion under probability measure $\mathbb{P}$:

$$\mathrm{d}X_t = \sigma\,\mathrm{d}W^{\mathbb{P}}, \quad \mathrm{d}W^{\mathbb{P}} \sim \mathrm{N}(0, \mathrm{d}t) \tag{H.5.1}$$

or

$$X_t = \sigma W_t^{\mathbb{P}}, \quad W_t^{\mathbb{P}} \sim \mathrm{N}(0, t)$$

Now we can choose another probability measure $\mathbb{Q}$ so that:

$$\mathrm{d}W^{\mathbb{P}} = \mathrm{d}W^{\mathbb{Q}} + \frac{\nu}{\sigma}\,\mathrm{d}t \tag{H.5.2}$$

where ν is a constant.

Under probability measure $\mathbb{Q}$ the motion in Eq. (H.5.1) is:

$$\mathrm{d}\overline{X}_t = \sigma\left(\mathrm{d}W^{\mathbb{Q}} + \frac{\nu}{\sigma}\mathrm{d}t\right) \tag{H.5.3}$$

so

$$\mathrm{d}\overline{X}_t = \nu\,\mathrm{d}t + \sigma\,\mathrm{d}W^{\mathbb{Q}} \tag{H.5.4}$$

It can be seen from Section 2.4 that the transformation between measures $\mathbb{P}$ and $\mathbb{Q}$ can be accomplished using $k = \nu/\sigma$, and that the associated Radon–Nikodym derivative is:

$$\begin{aligned}\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} &= \exp\left(kW_t^{\mathbb{P}} - \frac{1}{2}k^2t\right)\\ &= \exp\left(\frac{\nu}{\sigma}W_t^{\mathbb{P}} - \frac{1}{2}\frac{\nu^2 t}{\sigma}\right)\\ &= \exp\left(\frac{\nu}{\sigma^2}X_t - \frac{1}{2}\frac{\nu^2 t}{\sigma}\right)\end{aligned} \tag{H.5.5}$$

where we have used the fact that under probability measure $\mathbb{P}$ we can write: $W_t^{\mathbb{P}} = X_t/\sigma$.

Now

$$P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\right) = E^{\mathbb{Q}}\left[\mathbb{I}_{\{m_t^{\overline{X}} \leqslant b\}}\mathbb{I}_{\{\overline{X}_t \geqslant x\}}\right] \tag{H.5.6}$$

where $\mathbb{I}_{\{condition\}}$ is an indicator function which takes unit value when *condition* is satisfied and zero otherwise—for example $\mathbb{I}_{\{m_t^{\overline{X}} \leqslant b\}}$ is one when $m_t^{\overline{X}} \leqslant b$ and zero when $m_t^{\overline{X}} > b$.

However (see for example Baxter and Rennie (1996)), we have:

$$E^{\mathbb{Q}}\left[\mathbb{I}_{\{m_t^{\overline{X}} \leqslant b\}}\mathbb{I}_{\{\overline{X}_t \geqslant x\}}\right] = E^{\mathbb{P}}\left[\mathbb{I}_{\{m_t^{X} \leqslant b\}}\mathbb{I}_{\{X_t \geqslant x\}}\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right] \tag{H.5.7}$$

So substituting for $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}$ from Eq. (H.5.5) gives:

$$E^{\mathbb{Q}}\left[\mathbb{I}_{\{m_t^{\overline{X}} \leqslant b\}}\mathbb{I}_{\{\overline{X}_t \geqslant x\}}\right] = E^{\mathbb{P}}\left[\mathbb{I}_{\{m_t^{X} \leqslant b\}}\mathbb{I}_{\{X_t \geqslant x\}}\exp\left(\frac{\nu X_t}{\sigma^2} - \frac{\nu^2 t}{2\sigma^2}\right)\right] \tag{H.5.8}$$

Expressed in terms of the *reflected Brownian motion*, $X_t^R = 2b - X_t$, Eq. (H.5.8) can be written:

$$\begin{aligned} &P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) \\ &\quad = E^{\mathbb{P}}\left[\mathbb{I}_{\{2b-X_t^R \geqslant x\}} \exp\left(\frac{\nu(2b - X_t^R)}{\sigma^2} - \frac{\nu^2 t}{2\sigma^2}\right)\right] \\ &\quad = \exp\left(\frac{2\nu b}{\sigma^2}\right) E^{\mathbb{P}}\left[\mathbb{I}_{\{2b-X_t \geqslant x\}} \exp\left(-\frac{\nu X_t^R}{\sigma^2} - \frac{\nu^2 t}{2\sigma^2}\right)\right] \end{aligned} \tag{H.5.9}$$

Since

$$\mathbb{I}_{\{2b-X_t^R > x\}} = \mathbb{I}_{\{-2b+X_t^R < -x\}} = \mathbb{I}_{\{X_t^R < 2b-x\}}$$

Equation (H.5.9) becomes:

$$\begin{aligned} &P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) \\ &\quad = \exp\left(\frac{2\nu b}{\sigma^2}\right) E^{\mathbb{P}}\left[\mathbb{I}_{\{X_t < 2b-x\}} \exp\left(-\frac{\nu X_t}{\sigma^2} - \frac{\nu^2 t}{2\sigma^2}\right)\right] \end{aligned} \tag{H.5.10}$$

where, for ease of notation, we now denote X_t^R by X_t on the right-hand side of Eq. (H.5.10).

Therefore:

$$\begin{aligned} &P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) \\ &\quad = \exp\left(\frac{2\nu b}{\sigma^2}\right) \int_{X_t=-\infty}^{X_t=2b-x} \frac{1}{\sigma\sqrt{2\pi t}} \exp\left(-\frac{X_t^2}{2\sigma^2 t}\right) \exp\left(\frac{\nu X_t}{\sigma^2} - \frac{\nu^2 t}{\sigma^2}\right) \mathrm{d}X_t \\ &\quad = \frac{1}{\sigma\sqrt{2\pi t}} \exp\left(\frac{2\nu b}{\sigma^2}\right) \int_{X_t=-\infty}^{X_t=2b-x} \exp\left(-\frac{(X_t^2 + 2\nu X_t t + \nu^2 t^2)}{2\sigma^2 t}\right) \mathrm{d}X_t \\ &\quad = \frac{1}{\sigma\sqrt{2\pi t}} \exp\left(\frac{2\nu b}{\sigma^2}\right) \int_{X_t=-\infty}^{X_t=2b-x} \exp\left(-\frac{(X_t + \nu t)^2}{2\sigma^2 t}\right) \mathrm{d}X_t \end{aligned}$$

If $V = (X_t + \nu t)/(\sigma\sqrt{t})$ then $\mathrm{d}X_t = \sigma\sqrt{t}\,\mathrm{d}V$, $X_t = 2b - x$ corresponds to $V = (2b - x + \nu t)/(\sigma\sqrt{t})$, and $X_t = -\infty$ corresponds to $V = -\infty$.

$$\begin{aligned} &\frac{1}{\sigma\sqrt{2\pi t}} \int_{X_t=-\infty}^{X_t=2b-x} \exp\left(-\frac{(X_t + \nu t)^2}{2\sigma^2 t}\right) \mathrm{d}X_t \\ &\quad = \frac{1}{\sigma\sqrt{2\pi t}} \int_{V=-\infty}^{V=(2b-x+\nu t)/(\sigma\sqrt{t})} \exp\left(-\frac{V^2}{2}\right) \mathrm{d}V \\ &\quad = N_1\left(\frac{2b - x + \nu t}{\sigma\sqrt{t}}\right) \end{aligned}$$

We thus obtain:

$$P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) = \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{2b - x + \nu t}{\sigma\sqrt{t}}\right)$$

H.6 Proof of Eq. (H.1.7)

Since $P(\overline{X}_t \geqslant x) = P(m_t^{\overline{X}} \geqslant b, \overline{X}_t \geqslant x) + P(m_t^{\overline{X}} \geqslant b, \overline{X}_t \leqslant x)$ we have:

$$P\left(m_t^{\overline{X}} \geqslant b, \overline{X}_t \geqslant x\right) = P(\overline{X}_t \geqslant x) - P\left(m_t^{\overline{X}} \geqslant b, \overline{X}_t \leqslant x\right) \tag{H.6.1}$$

Substituting the results of Eqs. (H.1.4) and (H.1.6) into Eq. (H.6.1) yields:

$$P\left(m_t^{\overline{X}} \geqslant b, \overline{X}_t \geqslant x\right) = N_1\left(\frac{\nu t - x}{\sigma\sqrt{t}}\right) - \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{2b - x + \nu t}{\sigma\sqrt{t}}\right)$$

H.7 Proof of Eq. (H.1.8)

We start by writing:

$$P\left(m_t^{\overline{X}} \leqslant b\right) = P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \leqslant b\right) + P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant b\right)$$

However, $P(m_t^{\overline{X}} \leqslant b, \overline{X}_t \leqslant b) = P(\overline{X}_t \leqslant b)$ since the probability that the minimum is less than b and the final value $\overline{X}_t$ is less than b is the same as the probability that the final value $\overline{X}_t$ is less than b. Therefore:

$$P\left(m_t^{\overline{X}} \leqslant b\right) = P(\overline{X}_t \leqslant b) + P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant b\right)$$

Substituting for $P(\overline{X}_t \leqslant b)$ from Eq. (H.1.4) gives:

$$P\left(m_t^{\overline{X}} \leqslant b\right) = N_1\left(\frac{b - \nu t}{\sigma\sqrt{t}}\right) + P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant b\right) \tag{H.7.1}$$

From Eq. (H.1.6):

$$\begin{aligned} P\left(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant b\right) &= \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{2b - b + \nu t}{\sigma\sqrt{t}}\right) \\ &= \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{b + \nu t}{\sigma\sqrt{t}}\right) \end{aligned} \tag{H.7.2}$$

Combining Eqs. (H.7.2) and (H.7.1) yields:

$$P\left(m_t^{\overline{X}} \leqslant b\right) = N_1\left(\frac{b - \nu t}{\sigma\sqrt{t}}\right) + \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{b + \nu t}{\sigma\sqrt{t}}\right)$$

H.8 Proof of Eq. (H.1.9)

We start with:

$$P\left(m_t^{\overline{X}} \geqslant b\right) = 1 - P\left(m_t^{\overline{X}} \leqslant b\right)$$

Substituting from (H.1.8):

$$P\big(m_t^{\overline{X}} \geqslant b\big) = 1 - N_1\left(\frac{b-\nu t}{\sigma\sqrt{t}}\right) + \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{b+\nu t}{\sigma\sqrt{t}}\right) \tag{H.8.1}$$

But since $1 - N_1(x) = N_1(-x)$, Eq. (H.8.1) can be expressed as:

$$P\big(m_t^{\overline{X}} \geqslant b\big) = N_1\left(\frac{\nu t - b}{\sigma\sqrt{t}}\right) + \exp\left(\frac{2\nu b}{\sigma^2}\right) N_1\left(\frac{b+\nu t}{\sigma\sqrt{t}}\right)$$

H.9 Proof of Eq. (H.1.10)

We will use Eq. (H.1.6) to compute $\frac{\partial P}{\partial x}$, where $P(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x)$ is denoted by P.

Letting $\Theta = (2b - x + \nu t)/(\sigma\sqrt{t})$ we obtain:

$$\begin{aligned}
\frac{\partial P}{\partial x} &= \exp\left(\frac{2\nu b}{\sigma^2}\right) \frac{\partial}{\partial x}\big\{N_1(\Theta)\big\} \\
&= \exp\left(\frac{2\nu b}{\sigma^2}\right) \frac{\partial}{\partial \Theta}\big\{N_1(\Theta)\big\} \frac{\partial \Theta}{\partial x} \\
&= -\exp\left(\frac{2\nu b}{\sigma^2}\right) \frac{1}{\sigma\sqrt{t}} n(\Theta) \\
&= -\frac{1}{\sigma\sqrt{2\pi t}} \exp\left(\frac{2\nu b}{\sigma^2}\right) \exp\left(\frac{(2b - x + \nu t)^2}{2\sigma^2 t}\right)
\end{aligned} \tag{H.9.1}$$

Now since the probability $P(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x)$ decreases as x increases we have:

$$P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) - P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x + \Delta x\big) = -\frac{\partial P}{\partial x}\Delta x \tag{H.9.2}$$

and also:

$$\begin{aligned}
&P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x\big) - P\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x + \Delta x\big) \\
&\quad \sim p\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t = x\big)\Delta x
\end{aligned} \tag{H.9.3}$$

where $p(m_t^{\overline{X}} \leqslant b, \overline{X}_t = x)$ is the probability density function of $P(m_t^{\overline{X}} \leqslant b, \overline{X}_t \geqslant x)$, and $\Delta x \to 0$.

Combining Eqs. (H.9.2) and (H.9.3) we thus obtain

$$p\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t = x\big) = -\frac{\partial P}{\partial x}\Delta x$$

So,

$$p\big(m_t^{\overline{X}} \leqslant b, \overline{X}_t = x\big) = \frac{1}{\sigma\sqrt{2\pi t}} \exp\left(\frac{2\nu b}{\sigma^2}\right) \exp\left(-\frac{(2b - x + \nu t)^2}{2\sigma^2 t}\right)$$

which means that:

$$p\left(\left\{m_t^{\overline{X}} \leqslant b, \overline{X}_t\right\}|\overline{X}_0\right) = \frac{1}{\sigma\sqrt{2\pi t}}\exp\left(\frac{2\nu b}{\sigma^2}\right) \times \exp\left(-\frac{(2b - \overline{X}_t + \nu t)^2}{2\sigma^2 t}\right) \tag{H.9.4}$$

where as usual we take $\overline{X}_0 = 0$. So Eq. (H.9.4) gives the probability density for the Brownian motion which goes through the points $\overline{X}_0$ and $\overline{X}_t$ and has a minimum value which is less than or equal to b.

Instead of considering the complete path of $\overline{X}_t$ from $\overline{X}_0$ we can move the origin to the point $\overline{X}_{t_1}$, where $t_1 \leqslant t$. Substituting into Eq. (H.9.4) we then obtain:

$$\begin{aligned} p\left(\left\{m_{t_1,t}^{\overline{X}} \leqslant b, \overline{X}_t\right\}|\overline{X}_{t_1}\right) &= \frac{1}{\sigma\sqrt{2\pi(t-t_1)}}\exp\left(\frac{2\nu(b-\overline{X}_{t_1})}{\sigma^2}\right) \\ &\quad \times \exp\left(-\frac{(2(b-\overline{X}_{t_1}) - (\overline{X}_t - \overline{X}_{t_1}) + \nu(t-t_1))^2}{2\sigma^2(t-t_1)}\right) \\ &= \frac{1}{\sigma\sqrt{2\pi(t-t_1)}}\exp\left(\frac{2\nu(b-\overline{X}_{t_1})}{\sigma^2}\right) \\ &\quad \times \exp\left(-\frac{(2b - \overline{X}_{t_1} - \overline{X}_t + \nu(t-t_1))^2}{2\sigma^2(t-t_1)}\right) \end{aligned}$$

which can be re-expressed as:

$$p\left(\left\{m_{t_1,t_2}^{\overline{X}} \leqslant b, \overline{X}_{t_2}\right\}|\overline{X}_{t_1}\right) = \frac{1}{\sigma\sqrt{2\pi\Delta t}}\exp\left(\frac{2\nu(b-\overline{X}_{t_1})}{\sigma^2}\right) \times \exp\left(-\frac{(\overline{X}_{t_1} + \overline{X}_{t_2} - 2b - \nu\Delta t)^2}{2\sigma^2\Delta t}\right)$$

where $t_2 \geqslant t_1$ and $\Delta t = t_2 - t_1$.

Appendix I:
The Feynman–Kac formula

The Feynman–Kac formula provides a link between stochastic processes and partial differential equations, which we will now illustrate.

In the risk neutral measure the equation followed by the asset price is:

$$\mathrm{d}S = rS\,\mathrm{d}t + \sigma S\,\mathrm{d}W \tag{I.1.1}$$

and that of the money account:

$$\mathrm{d}B = Br\,\mathrm{d}t \tag{I.1.2}$$

If $f(S,t)$ is the value of a derivative then using Ito's lemma we have:

$$\mathrm{d}f = \left\{\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2}\right\} + \frac{\partial f}{\partial S}\sigma\,\mathrm{d}W \tag{I.1.3}$$

Since f is a tradable we know that the process $(\frac{f}{B})$ must be a martingale in the risk neutral measure, and therefore have zero drift.

We will now evaluate $\mathrm{d}(\frac{f}{B})$ using the Ito quotient rule (see Eq. (2.6.4)):

$$\begin{aligned}\mathrm{d}\left(\frac{X_1}{X_2}\right) = \left(\frac{X_1}{X_2}\right)\left\{\frac{\mathrm{d}X_1}{X_1} - \frac{\mathrm{d}X_2}{X_2}\right\} + E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] \\ - E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_1}{X_1}\right)\right]\end{aligned} \tag{I.1.4}$$

and rewrite Eqs. (I.1.2) and (I.1.3) as:

$$\mathrm{d}X_1 = \bar{\mu}_1\,\mathrm{d}t + \bar{\sigma}_1\,\mathrm{d}W$$

$$\mathrm{d}X_2 = X_2\bar{\mu}_2\,\mathrm{d}t$$

where

$$\mathrm{d}\left(\frac{X_1}{X_2}\right) = \mathrm{d}\left(\frac{f}{B}\right), \qquad \bar{\mu}_1 = \left\{\frac{\partial f}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2}\right\}$$

$$\bar{\sigma}_1 = \sigma_1\frac{\partial f}{\partial S}, \qquad X_1 = f, \qquad X_2 = B, \qquad \bar{\mu}_2 = r$$

Evaluating Eq. (I.1.4) we obtain:

$$E\left[\left(\frac{\mathrm{d}X_2}{X_2}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] = E\left[\bar{\mu}_2^2\,\mathrm{d}t^2\right] \to 0$$

$$E\left[\left(\frac{\mathrm{d}X_1}{X_1}\right)\left(\frac{\mathrm{d}X_2}{X_2}\right)\right] = E\left[\left(\frac{\bar{\mu}_1\,\mathrm{d}t + \bar{\sigma}_1\,\mathrm{d}W}{X_1}\right)\left(\frac{X_2\bar{\mu}_2\,\mathrm{d}t}{X_2}\right)\right] \to 0$$

and therefore:

$$\begin{aligned} \mathrm{d}\left(\frac{X_1}{X_2}\right) &= \left(\frac{X_1}{X_2}\right)\left\{\frac{\mu_1\,\mathrm{d}t + \bar{\sigma}_1\,\mathrm{d}W}{X_1} - \frac{X_2\bar{\mu}_2\,\mathrm{d}t}{X_2}\right\} \\ &= \left\{\frac{\bar{\mu}_1}{X_2} - \left(\frac{X_1}{X_2}\right)\bar{\mu}_2\right\}\mathrm{d}t + \left(\frac{\bar{\sigma}_1}{X_2}\right) \\ &= \frac{1}{X_2}\{\bar{\mu}_1 - X_1\bar{\mu}_2\}\,\mathrm{d}t + \left(\frac{\bar{\sigma}_1}{X_2}\right) \end{aligned} \tag{I.1.5}$$

Since $(\frac{X_1}{X_2})$ is a martingale, the drift term in Eq. (I.1.5) is zero so:

$$\bar{\mu}_1 - X_1\bar{\mu}_2 = 0 \tag{I.1.6}$$

Therefore, substituting for $\bar{\mu}_1$, X_1 and $\bar{\mu}_2$ in Eq. (I.1.6) we obtain:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2} - rf = 0 \tag{I.1.7}$$

or

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2} = rf \tag{I.1.8}$$

which is the Black–Scholes partial differential equation, which we derived in Chapter 4.

In general if an asset follows the process:

$$\mathrm{d}S = \bar{\mu}\,\mathrm{d}t + \bar{\sigma}\,\mathrm{d}W \tag{I.1.9}$$

then the price of a derivative $f(S,t)$ obeys the partial differential equation:

$$\frac{\partial f}{\partial t} + \bar{\mu}\frac{\partial f}{\partial S} + \frac{\bar{\mu}^2}{2}\frac{\partial^2 f}{\partial S^2} = rf \tag{I.1.10}$$

or

$$\left(\frac{\partial}{\partial t} + \bar{\mu}\frac{\partial}{\partial S} + \frac{\bar{\mu}^2}{2}\frac{\partial^2}{\partial S^2}\right)f = rf \tag{I.1.11}$$

Appendix J: Answers to problems

Problem 1

Let $\beta_t^k = E[W_t^k]$, where $W_{t_0} = 0$.

(a) Show using Ito's formula for $k = 2, 3, 4, \ldots$ that

$$\beta_t^k = \frac{1}{2}(k-1)\int_{s=0}^{t} \beta_s^{k-2}\,\mathrm{d}s$$

(b) Deduce that $E[W_t^4] = 3t^2$
(c) What is $E[W_t^6]$

1(a)

Let $\phi(W_t) = W_t^k$; using Ito's formula we have:

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial W_t}\,\mathrm{d}W_t + \frac{1}{2}\frac{\partial^2 \phi}{\mathrm{d}W_t^2}\,\mathrm{d}t$$

So

$$\mathrm{d}\left(W_t^k\right) = kW_t^{k-1}\,\mathrm{d}W_t + \frac{1}{2}k(k-1)W_t^{k-2}\,\mathrm{d}t$$

Integrating both sides

$$\int_{s=0}^{t} \mathrm{d}\left(W_s^k\right) = k\int_{s=0}^{t} W_s^{k-1}\,\mathrm{d}W_s + \frac{1}{2}(k-1)\int_{s=0}^{t} W_s^{k-2}\,\mathrm{d}s$$

$$W_t^k - W_{t_0}^k = k\int_{s=0}^{t} W_s^{k-1}\,\mathrm{d}W_s + \frac{1}{2}(k-1)\int_{s=0}^{t} W_s^{k-2}\,\mathrm{d}s$$

Now

$$E\left[\int_{s=0} W_s^{k-1}\,\mathrm{d}W_s\right] = 0$$

and using Fubini's theorem

$$E\left[\int_{s=0}^{t} W_s^{k-2}\,\mathrm{d}s\right] = \int_{s=0}^{t} E\left[W_s^{k-2}\right]\mathrm{d}s$$

Therefore

$$E\big[W_t^k\big] - E\big[W_{t_0}^k\big] = kE\left[\int_{s=0}^{t} W_s^{k-1}\,\mathrm{d}W_s\right] + \frac{1}{2}(k-1)\int_{s=0}^{t} E\big[W_s^{k-2}\big]\,\mathrm{d}s$$
$$= \frac{1}{2}(k-1)\int_{s=0}^{t} E\big[W_s^{k-2}\big]\,\mathrm{d}s$$

Since $E[W_{t_0}^k] = 0$ we obtain

$$\beta_t^k = \frac{1}{2}(k-1)\int_{s=0}^{t} \beta_s^{k-2}\,\mathrm{d}s$$

1(b)

Since W_t is standard Brownian motion (zero drift and $\sigma = 1$)

$$E\big[W_t^2\big] = t$$

Substituting $k = 4$ in the relation from part (a) yields

$$E\big[W_t^4\big] = \frac{4\times 3}{2}\int_{s=0}^{t} E\big[W_s^2\big]\,\mathrm{d}s = \frac{12}{2}\int_{s=0}^{t} s\,\mathrm{d}s = 6\frac{t^2}{2} = 3t^2$$

1(c)

Using the relation from part (a) with $k = 6$ yields

$$E\big[W_t^6\big] = \frac{6\times 5}{2}\int_{s=0}^{t} E\big[W_s^4\big]\,\mathrm{d}s = \frac{6\times 5}{2}\int_{s=0}^{t} 3s^2\,\mathrm{d}s = \frac{30}{2}\frac{3t^3}{3} = 15t^3$$

Problem 2

Solve

$$\mathrm{d}X_t = X_t\,\mathrm{d}t + \mathrm{d}W_t$$

Rearranging we have

$$\mathrm{d}X_t - X_t\,\mathrm{d}X_t = \mathrm{d}W_t$$

Using the integrating factor $\exp(-t)$ gives

$$\exp(-t)\,\mathrm{d}X_t - X_t\exp(-t)\,\mathrm{d}t = \exp(-t)\,\mathrm{d}W_t$$

and

$$\mathrm{d}\big(X_t\exp(-t)\big) = -X_t\exp(-t)\,\mathrm{d}t + \exp(-t)\,\mathrm{d}X_t = \exp(-t)\,\mathrm{d}W_t$$

So

$$\mathrm{d}\big(X_t\exp(-t)\big) = \exp(-t)\,\mathrm{d}W_t$$

Integrating both sides

$$\int_{s=0}^{t} \mathrm{d}\big(X_s \exp(-s)\big) = \int_{s=0}^{t} \exp(-s)\,\mathrm{d}W_s$$

and

$$X_t \exp(-t) - X_{t_0} = \int_{s=0}^{t} \exp(-s)\,\mathrm{d}W_s$$

which means that

$$X_t = X_{t_0} \exp(t) + \int_{s=0}^{t} \exp(t-s)\,\mathrm{d}W_s$$

Problem 3

Solve

$$\mathrm{d}X_t = -X_t\,\mathrm{d}t + \exp(-t)\,\mathrm{d}W_t$$

Rearranging yields

$$\mathrm{d}X_t + X_t\,\mathrm{d}t = \exp(-t)\,\mathrm{d}W_t$$

Using the integrating factor $\exp(t)$ we obtain

$$\exp(t)\,\mathrm{d}X_t + \exp(t)X_t\,\mathrm{d}t = \mathrm{d}W_t$$

and

$$\mathrm{d}\big(\exp(t)X_t\big) = \exp(t)\,\mathrm{d}X_t + \exp(t)X_t\,\mathrm{d}t$$

which means that

$$\mathrm{d}\big(\exp(t)X_t\big) = \mathrm{d}W_t \quad \text{and} \quad \int_{s=0}^{t} \mathrm{d}\big(\exp(s)X_s\big) = \int_{s=0}^{t} \mathrm{d}W_s$$

Integrating

$$X_t \exp(t) - X_{t_0} = W_t - W_{t_0}$$

Since $W_{t_0} = 0$

$$X_t \exp(t) - X_{t_0} = W_t$$

that is

$$X_t = X_{t_0} \exp(-t) + W_t \exp(-t)$$

Problem 4

Prove

$$\int_{s=0} W_s^2 \, \mathrm{d}W_s = \frac{1}{3} W_t^3 - \int_{s=0}^{t} W_s \, \mathrm{d}s$$

Using Ito's formula

$$\mathrm{d}\left(W_t^3\right) = 3W_t^2 \, \mathrm{d}W + \frac{6}{2} W_t \, \mathrm{d}t$$

Therefore

$$\int_{s=0}^{t} \mathrm{d}\left(W_t^3\right) = 3 \int_{s=0}^{t} W_s^2 \, \mathrm{d}W_s + 3 \int_{s=0}^{t} W_s \, \mathrm{d}s$$

and

$$W_t^3 - W_{t_0}^3 = 3 \int_{s=0}^{t} W_s^2 \, \mathrm{d}W_s + 3 \int_{s=0}^{t} W_s \, \mathrm{d}s$$

Using $W_{t_0} = 0$ we obtain

$$W_t^3 = 3 \int_{s=0}^{t} W_s^2 \, \mathrm{d}W_s + 3 \int_{s=0}^{t} W_s \, \mathrm{d}s$$

So

$$\int_{s=0}^{t} W_s^2 \, \mathrm{d}W_s = \frac{1}{3} W_t^3 - \int_{s=0}^{t} W_s \, \mathrm{d}s$$

Problem 5

Solve $\mathrm{d}Y_t = r \, \mathrm{d}t + \alpha Y_t \, \mathrm{d}W_t$ where r and α are real constants.

Use the integrating factor $F_t = \exp(-\alpha W_t + \frac{\alpha^2}{2} t)$

$$\mathrm{d}Y_t - \alpha Y_t \, \mathrm{d}t = r \, \mathrm{d}t$$

Multiplying by F_t

$$\mathrm{d}Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) - \alpha Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) = r \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) \mathrm{d}t$$

Using Ito's formula

$$\mathrm{d}\left(Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right)\right) = \mathrm{d}Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) - \alpha Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) \mathrm{d}W_t$$

$$= r \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) \mathrm{d}t$$

Integrating

$$\int_{s=0}^{t} \mathrm{d}\left(Y_s \exp\left(-\alpha W_s + \frac{\alpha^2}{2} s\right)\right) = r \int_{s=0}^{t} \exp\left(-\alpha W_s + \frac{\alpha^2}{2} s\right) \mathrm{d}s$$

$$Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) - Y_{t_0} \exp(-\alpha W_{t_0}) = r \int_{s=0}^{t} \exp\left(-\alpha W_s + \frac{\alpha^2}{2} s\right) \mathrm{d}s$$

Using $W_{t_0} = 0$ yields

$$Y_t \exp\left(-\alpha W_t + \frac{\alpha^2}{2} t\right) = Y_{t_0} + r \int_{s=0}^{t} \exp\left(-\alpha W_s + \frac{\alpha^2}{2} s\right) \mathrm{d}s$$

So

$$Y_t = Y_{t_0} \exp\left(\alpha W_t - \frac{\alpha^2}{2} t\right) + r \int_{s=0}^{t} \exp\left(\alpha (W_t - W_s) + \frac{\alpha^2}{2} (t - s)\right) \mathrm{d}s$$

Problem 6

6(a)

Solve

$$\mathrm{d}X_t = (m - X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$$

where m and σ are constants.

Rearranging

$$\mathrm{d}X_t + X_t\,\mathrm{d}t = m\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$$

Use the integrating factor $\exp(t)$

$$\begin{aligned} \mathrm{d}\big(X_t \exp(t)\big) &= \exp(t)\,\mathrm{d}X_t + X_t \exp(t)\,\mathrm{d}t \\ &= m \exp(t)\,\mathrm{d}t + \sigma \exp(t)\,\mathrm{d}W_t \end{aligned}$$

Integrating

$$\int_{s=0}^{t} \mathrm{d}\big(X_s \exp(s)\big) = m \int_{s=0}^{t} \exp(s)\,\mathrm{d}s + \sigma \int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s$$

$$X_t \exp(t) - X_{t_0} = m\big(\exp(t) - 1\big) + \sigma \int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s$$

which can be expressed as

$$X_t = m + (X_{t_0} - m) \exp(-t) + \sigma \exp(-t) \int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s$$

6(b)

Taking expectations of the expression for X_t derived in part (a)

$$E[X_t] = m + (X_{t_0} - m)\exp(-t)$$

where we have used

$$E\left[\int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s\right] = 0$$

$$\begin{aligned}\mathrm{Var}[X_t] &= E\left[\left(X_t - E[X_t]\right)^2\right] \\ &= E\left[\sigma^2 \exp(-2t)\left\{\int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s\right\}^2\right] \\ &= \sigma^2 \exp(-2t) E\left[\left\{\int_{s=0}^{t} \exp(s)\,\mathrm{d}W_s\right\}^2\right]\end{aligned}$$

From Ito's isometry

$$\begin{aligned}\mathrm{Var}[X_t] &= \sigma^2 \exp(-2t) E\left[\int_{s=0}^{t} \exp(2s)\,\mathrm{d}s\right] \\ &= \sigma^2 \exp(-2t)\left[\frac{\exp(2s)}{2}\right]_{s=0}^{t} \\ &= \sigma^2 \exp(-2t)\left\{\frac{\exp(2t)}{2} - \frac{1}{2}\right\}\end{aligned}$$

which gives

$$\mathrm{Var}[X_t] = \frac{1}{2\sigma^2}\left\{1 - \exp(-2t)\right\}$$

Problem 7

Consider the equation $\mathrm{d}S_t = \mu_t S_t\,\mathrm{d}t + \sigma_t S_t\,\mathrm{d}W_t$ where the value of S_t at time $t = 0$ is denoted by S_0.

(a) Show that the mean is

$$E\left[\log(S_t)\right] = \log(S_0) + \int_{\tau=0}^{t} \left\{\mu_\tau - \frac{\sigma_\tau^2}{2}\right\} \mathrm{d}\tau$$

(b) Show that the variance is

$$\mathrm{Var}\left[\log(S_t)\right] = \int_{\tau=0}^{t} \sigma_\tau^2\,\mathrm{d}\tau$$

7(a)

If $\phi = \log(S)$ then using Ito's formula we have

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial S}\,\mathrm{d}S + \frac{1}{2}\frac{\partial^2 \phi}{\partial S^2}E\big[(\mathrm{d}S)^2\big] = \frac{1}{S}\{\mu_t S_t\,\mathrm{d}t + \sigma_t S_t\,\mathrm{d}W_t\} - \frac{1}{2}\frac{S^2}{S^2}\sigma_t\,\mathrm{d}t$$

So

$$\mathrm{d}\big(\log(S_t)\big) = \left(\mu_t - \frac{\sigma_t^2}{2}\right)\mathrm{d}t + \sigma_t\,\mathrm{d}W_t$$

and therefore

$$\int_{\tau=0}^{t}\mathrm{d}\big(\log(S_\tau)\big) = \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau + \int_{\tau=0}^{t}\sigma_\tau\,\mathrm{d}W_\tau$$

which gives

$$\log(S_\tau) - \log(S_0) = \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau + \int_{\tau=0}^{t}\sigma_\tau\,\mathrm{d}W_\tau$$

Taking expectations we obtain

$$E\big[\log(S_t)\big] - E\big[\log(S_0)\big] = E\left[\int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau\right] + E\left[\int_{\tau=0}^{t}\sigma_\tau\,\mathrm{d}W_\tau\right]$$

Since $\int_{\tau=0}^{t}(\mu_\tau - \frac{\sigma_\tau^2}{2})\,\mathrm{d}\tau$ is deterministic

$$E\left[\int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau\right] = \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau$$

and using

$$E\left[\int_{\tau=0}^{t}\sigma_\tau\,\mathrm{d}W_\tau\right] = 0$$

and

$$E\big[\log(S_0)\big] = \log(S_0)$$

we finally obtain

$$E\big[\log(S_t)\big] = \log(S_0) + \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau$$

7(b)

$$\begin{aligned}\mathrm{Var}\big[\log(S_t)\big] &= E\big[\{\log(S_t) - E\big[\log(S_t)\big]\}^2\big] \\ &= E\left[\left\{\log(S_0) + \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau \right.\right. \\ &\qquad \left.\left. + \int_{\tau=0}^{t}\sigma_\tau\,\mathrm{d}W_\tau - \log(S_0) - \int_{\tau=0}^{t}\left(\mu_\tau - \frac{\sigma_\tau^2}{2}\right)\mathrm{d}\tau\right\}^2\right]\end{aligned}$$

$$= E\left[\left(\int_{\tau=0}^{t} \sigma_\tau \, \mathrm{d}W_\tau\right)^2\right]$$

Using Ito's isometry we have:

$$E\left[\left(\int_{\tau=0}^{t} \sigma_\tau \, \mathrm{d}\tau\right)^2\right] = E\left[\int_{\tau=0}^{t} \sigma_\tau^2 \, \mathrm{d}\tau\right]$$

Since $\int_{\tau=0}^{t} \sigma_\tau^2 \, \mathrm{d}\tau$ is deterministic we can write

$$E\left[\int_{\tau=0}^{t} \sigma_\tau^2 \, \mathrm{d}\tau\right] = \int_{\tau=0}^{t} \sigma_\tau^2 \, \mathrm{d}\tau$$

and finally we obtain

$$\mathrm{Var}\big[\log(S_t)\big] = \int_{\tau=0}^{t} \sigma_\tau^2 \, \mathrm{d}\tau$$

Problem 8

Prove that if $\phi = \exp(t W_t)$ then

$$\mathrm{d}\phi = \phi\left(W_t + \frac{t^2}{2}\right)\mathrm{d}t + t\phi \, \mathrm{d}W_t$$

From Ito we have

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \frac{\partial \phi}{\partial W_t}\,\mathrm{d}W_t + \frac{1}{2}\frac{\partial^2 \phi}{\partial W_t^2} E\big[(\mathrm{d}W_t)^2\big]$$

Now

$$\frac{\partial \phi}{\partial t} = W_t \exp(t W_t), \qquad \frac{\partial \phi}{\partial W_t} = t \exp(t W_t), \qquad \frac{\partial^2 \phi}{\partial W_t^2} = t^2 \exp(t W_t)$$

So

$$\mathrm{d}\phi = W_t \exp(t W_t)\,\mathrm{d}t + t \exp(t W_t)\,\mathrm{d}W_t + \frac{t^2}{2}\exp(t W_t)\,\mathrm{d}t$$

where we have used $E[(\mathrm{d}W_t)^2] = \mathrm{d}t$.

Therefore

$$\mathrm{d}\phi = \phi\left(W_t + \frac{t^2}{2}\right)\mathrm{d}t + t\phi \, \mathrm{d}W_t$$

Problem 9

Given

$$Z_t = \exp\left(\int_{s=0}^{t} \theta_s \, \mathrm{d}W_s - \frac{1}{2}\int_{s=0}^{t} \theta_s^2 \, \mathrm{d}s\right)$$

Use Ito to prove that the process for Z_t is $\mathrm{d}Z_t = Z_t\theta_t\,\mathrm{d}W_t$.

9(a)

Let

$$X_t = \int_{s=0}^{t} \theta_s\,\mathrm{d}W_s - \frac{1}{2}\int_{s=0}^{t} \theta_s^2\,\mathrm{d}s$$

so

$$\mathrm{d}X_t = \theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t$$

We thus have:

$$Z_t = \exp(X_t)$$

Using Ito we have

$$\mathrm{d}Z_t = \frac{\partial Z_t}{\partial X_t}\,\mathrm{d}X_t + \frac{\partial^2 Z_t}{\partial X_t^2}(\mathrm{d}X_t)^2$$

so

$$\mathrm{d}Z_t = Z_t\left\{\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right\} + Z_t E\left[\left(\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right)\left(\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right)\right]$$

Now

$$E\left[\left(\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right)\left(\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right)\right] = E\left[\theta_t^2\,\mathrm{d}W_t^2\right] + E\left[\frac{1}{4}\theta_t^4\,\mathrm{d}t^2\right] - E\left[\theta_t^3\,\mathrm{d}W_t\,\mathrm{d}t\right]$$

Ignoring terms in $\mathrm{d}t$ of order higher than 1 using the fact that:

$$E\left[\mathrm{d}W_t^2\right] = \theta_t^2 E\left[\mathrm{d}W_t^2\right] = \theta_t^2\,\mathrm{d}t \quad \text{and} \quad E\left[\theta_t^3\theta_t\,\mathrm{d}W_t\,\mathrm{d}t\right] = \theta_t^3\,\mathrm{d}t\,E[\mathrm{d}W_t] = 0$$

We have

$$\mathrm{d}Z_t = Z_t\left\{\theta_t\,\mathrm{d}W_t - \frac{1}{2}\theta_t^2\,\mathrm{d}t\right\} + \frac{1}{2}Z_t\theta_t^2\,\mathrm{d}t = Z_t\theta_t\,\mathrm{d}W_t$$

Hence we have shown that

$$\mathrm{d}Z_t = Z_t\theta_t\,\mathrm{d}W_t$$

Problem 10

Let $S_t = S_0 \exp(\mu t + \sigma W_t)$ where μ and σ are constants.

(a) Show by Ito's lemma that

$$\mathrm{d}S_t = \left(\mu + \frac{\sigma^2}{2}\right) S_t \,\mathrm{d}t + \sigma S_t \,\mathrm{d}W_t$$

(b) Show that

$$E[S_t] - E[S_0] = \left(\mu + \frac{\sigma^2}{2}\right) \int_{\tau=0}^{t} E\big[S(\tau)\big] \,\mathrm{d}\tau$$

(c) Show that

$$E[S_t] = S_0 \exp\left(\mu t + \frac{\sigma^2}{2} t\right)$$

10(a)

Let $\phi = S_0 \exp(\mu t + \sigma W_t)$.

Then using Ito's formula:

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial t} \,\mathrm{d}t + \frac{\partial \phi}{\partial W_t} \,\mathrm{d}W_t + \frac{1}{2} \frac{\partial^2 \phi}{\partial W_t^2} E\big[(\mathrm{d}W_t)^2\big]$$

So

$$\mathrm{d}\phi = \mu\phi \,\mathrm{d}t + \sigma\phi \,\mathrm{d}W_t + \frac{1}{2}\phi\sigma^2 \,\mathrm{d}t$$

where we have used $\frac{\partial^2 \phi}{\partial W_t^2} = \phi\sigma^2$ and $E[(\mathrm{d}W_t)] = \mathrm{d}t$.

Therefore

$$\mathrm{d}S_t = \left(\mu + \frac{1}{2}\sigma^2\right) S_t \,\mathrm{d}t + S_t \sigma \,\mathrm{d}W_t$$

10(b)

From part (a) we have

$$\int_{\tau=0}^{t} \mathrm{d}S_\tau = \left(\mu + \frac{1}{2}\sigma^2\right) \int_{\tau=0}^{t} S_\tau \,\mathrm{d}\tau + \sigma \int_{\tau=0}^{t} S_\tau \,\mathrm{d}W_\tau$$

Therefore

$$S_t - S_0 = \left(\mu + \frac{1}{2}\sigma^2\right) \int_{\tau=0}^{t} S_\tau \,\mathrm{d}\tau + \sigma \int_{\tau=0}^{t} S_\tau \,\mathrm{d}W_\tau$$

Taking expectations we have

$$E[S_t] - E[S_0] = \left(\mu + \frac{1}{2}\sigma^2\right) E\left[\int_{\tau=0}^{t} S_\tau \,\mathrm{d}\tau\right] + \sigma E\left[\int_{\tau=0}^{t} S_\tau \,\mathrm{d}W_\tau\right]$$

Using the fact that:

$$E\left[\int_{\tau=0}^{t} f(\tau)\,\mathrm{d}W_\tau\right] = 0$$

and from Fubini's theorem:

$$E\left[\int_{\tau}^{t} S(\tau)\,\mathrm{d}\tau\right] = \int_{\tau}^{t} E\big[S(\tau)\big]\,\mathrm{d}\tau$$

we thus finally obtain:

$$E[S_t] - E[S_0] = \left(\mu + \frac{1}{2}\sigma^2\right)\int_{\tau=0}^{t} E[S_\tau]\,\mathrm{d}\tau$$

10(c)

From part (a) we know:

$$\mathrm{d}S_t = \left(\mu + \frac{1}{2}\sigma^2\right)S_t\,\mathrm{d}t + S_t\sigma\,\mathrm{d}W_t$$

Therefore

$$\mathrm{d}\big(\log(S_t)\big) = \left(\mu + \frac{1}{2}\sigma^2\right)\mathrm{d}t + \sigma\,\mathrm{d}W_t$$

$$\int_{\tau=0}^{t} \mathrm{d}\big(\log(S_\tau)\big) = \left(\mu + \frac{1}{2}\sigma^2\right)\int_{\tau=0}^{t}\mathrm{d}\tau + \sigma\int_{\tau=0}^{t}\mathrm{d}W_\tau$$

So

$$\log(S_t) - \log(S_0) = \left(\mu + \frac{1}{2}\sigma^2\right)t + \sigma\int_{\tau=0}^{t}\mathrm{d}W_\tau$$

Taking expectations we obtain:

$$E\big[\log(S_t)\big] - E\big[\log(S_0)\big] = \left(\mu + \frac{1}{2}\sigma^2\right)t + E\left[\sigma\int_{\tau=0}^{t}\mathrm{d}W_\tau\right]$$

Since

$$E\left[\sigma\int_{\tau=0}^{t}\mathrm{d}W_\tau\right] = 0 \quad \text{and} \quad E[S_0] = S_0$$

we have:

$$\log\left(\frac{E[S_t]}{S_0}\right) = \left(\mu + \frac{1}{2}\sigma^2\right)t$$

which yields:

$$E[S_t] = S_0\exp\left\{\left(\mu + \frac{1}{2}\sigma^2\right)t\right\}$$

Problem 11

Let $\phi = X_t Y_t$.

From Ito's formula we obtain:

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial X_t}\,\mathrm{d}X_t + E\left[\frac{\partial Y_t}{\partial Y_t}\,\mathrm{d}Y_t + \frac{1}{2}\frac{\partial^2 \phi}{\partial X_t\,\partial Y_t}\,\mathrm{d}X_t\,\mathrm{d}Y_t + \frac{1}{2}\frac{\partial^2 \phi}{\partial Y_t\,\partial X_t}\,\mathrm{d}Y_t\,\mathrm{d}X_t\right]$$

Now

$$\frac{\partial^2 \phi}{\partial X_t\,\partial Y_t} = \frac{\partial^2 \phi}{\partial Y_t\,\partial X_t} = 1, \qquad \frac{\partial \phi}{\partial X_t} = Y_t \quad \text{and} \quad \frac{\partial \phi}{\partial Y_t} = X_t$$

We thus obtain:

$$\mathrm{d}\phi = Y_t\,\mathrm{d}X_t + X_t\,\mathrm{d}Y_t + E[\mathrm{d}X_t\,\mathrm{d}Y_t]$$

Using

$$\mathrm{d}(X_t Y_t) = Y_t\,\mathrm{d}X_t + X_t\,\mathrm{d}Y_t + E[\mathrm{d}X_t\,\mathrm{d}Y_t]$$

we have:

$$\int_{s=0}^{t} \mathrm{d}(X_t Y_t) = \int_{s=0}^{t} Y_s\,\mathrm{d}X_s + \int_{s=0}^{t} X_s \mathrm{d}Y_s + \int_{s=0}^{t} E[\mathrm{d}X_s\,\mathrm{d}Y_s]$$

Therefore

$$X_t Y_t - X_{t_0} Y_{t_0} = \int_{s=0}^{t} Y_s \mathrm{d}X_s + \int_{s=0}^{t} X_s \mathrm{d}Y_s + \int_{s=0}^{t} E[\mathrm{d}X_s\,\mathrm{d}Y_s]$$

Thus

$$\int_{s=0}^{t} X_s \mathrm{d}Y_s = X_t Y_t - X_{t_0} Y_{t_0} - \int_{s=0}^{t} Y_s \mathrm{d}X_s - \int_{s=0}^{t} E[\mathrm{d}X_s\,\mathrm{d}Y_s]$$

References

Abramowitz, M., and Stegun, I.A. (1968). *Handbook of Mathematical Functions*. Dover Publications.

Bachelier, L. (1900). Théorie de la spéculation. *Annales scientifiques de l'École Normale Supérieure*, 17:21–86.

Barone-Adesi, G., and Whaley, R.E. (1987). Efficient Analytic Approximation of American Option Values. *The Journal of Finance*, 42(2):301–320.

Barraquand, J., and Martineau, D. (1995). Numerical Valuation of High Dimensional Multivariate American Securities. *Journal of Finance ad Quantitative Analysis*, 30:383–405.

Baxter, M., and Rennie, A. (1996). *Financial Calculus: An Introduction to Derivative Pricing*. Cambridge University Press.

Black, F. (1973). Fact and Fantasy in the Use of Options and Corporate Liabilities. *Financial Analysts Journal*, 31:36–41, 61–72. *Journal of Political Economy*, 81:637–657.

Boyle, P.P., and Tian, Y. (1998). An Explicit Finite Difference Approach to the Pricing of Barrier Options. *Applied Mathematical Finance*, 5:17–43.

Box, G.E.P., and Muller, M.E. (1958). A Note on the Generation of Random Normal Deviates. *Annals of Mathematical Statistics*, 29:610–611.

Boyle, P.P., Evnine, J., and Gibbs, S. (1989). Numerical Evaluation of Multivariate Contingent Claims. *The Review of Management Studies*, 2(2):241–250.

Boyle, P.P., Broadie, M., and Glasserman, P. (1997). Monte Carlo Methods for Security Pricing. *Journal of Economic Dynamics and Control*, 21:1267–1321.

Brigo, D., and Mercurio, F. (2001). *Interest Rate Models – Theory and Practice: With Smile, Inflation and Credit*. Springer-Verlag.

Broadie, M., and DeTemple, J. (1996). American Option Valuation: New Bounds, Approximations, and a Comparison of Existing Methods. *The Review of Financial Studies*, 9(4):1211–1250.

Broadie, M., and Glasserman, P. (1997). Pricing American-Style Securities Simulation. *Journal of Economic Dynamics and Control*, 21:1323–1352.

Brotherton-Ratcliffe, R. (1994). Monte Carlo Motoring. *Risk*, 7(12):53–58.

Caflisch, R.E., Morokoff, W., and Owen, A. (1997). Valuation of Mortgage-Backed Securities Using Brownian Bridges to Reduce Effective Dimension. *The Journal of Computational Finance*, 1(1):27–46.

Cox, D.R., and Miller, H.D. (1965). *The Theory of Stochastic Processes*. Methuen & Co Ltd.

Cox, J.C., Ross, S.A., and Rubinstein, M. (1979). Option Pricing: A Simplified Approach. *Journal of Financial Economics*, 7:229–263.

Crank, J., and Nicolson, P. (1947). A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat Conduction Type. *Proceedings of the Cambridge Philosophical Society*, 43:50–67.

Einstein, A. (1905). On the Movement of Small Particles Suspended in a Stationary Liquid Demanded by the Molecular-Kinetic Theory of Heat. *Annalen der Physik*, 17.

Evans, M., Hastings, N., and Peacock, B. (2000). *Statistical Distributions*, Third Edition. Wiley.

Garman, M.B., and Kohlhagen, S.W. (1983). Foreign Currency Option Values. *Journal of International Money and Finance*, 2:231–237.

Geske, R. (1979). A Note on an Analytic Valuation Formula for Unprotected American Options on Stocks with Known Dividends. *Journal of Econometrics*, 7:375–380.

Geske, R., and Johnson, H.E. (1984). The American Put Options Valued Analytically. *Journal of Finance*, 39:1511–1524.

Golub, G.H., and Van Loan, C.F. (1989). *Matrix Computation*. The John Hopkins University Press.

Grimmett, G., and Welsh, D. (1986). *Probability: An Introduction*. Oxford Science Publications.

Hager, W. (1988). *Applied Numerical Linear Algebra*. Prentice Hall.

Harrison, J.M., and Kreps, D. (1979). Martingales and Arbitrage in Multiperiod Securities Markets. *Journal of Economic Theory*, 20:381–408.

Harrison, J.M., and Pliska, D. (1981). Martingales and Stochastic Integrals in the Theory of Continuous Trading. *Stochastic Processes and Their Applications*, 11:215–260.

Higham, N.J. (2002). Computing the Nearest Correlation Matrix – A Problem from Finance. *IMA Journal of Numerical Analysis*, 22(3):329–343.

Hull, J.C. (1997). *Options, Futures and Other Derivatives*, Third Edition. Prentice Hall.

Hull, J.C. (2003). *Options, Futures and Other Derivatives*, Fifth Edition. Prentice Hall.

Johnson, H. (1987). Options on the Maximum or the Minimum of Several Assets. *Journal of Financial and Quantitative Analysis*, 22(3):277–283.

Kamrad, B., and Ritchken, P. (1991). Multinomial Approximating Models for Options with *k* State Variables. *Management Science*, 37(12):1640–1652.

Karatzas, I., and Shreve, S. (1991). *Brownian Motion and Stochastic Calculus*. Springer-Verlag, New York.

Levy, P. (1939). Sur certains processus stochastiques homogènes. *Compositio Mathematica*, 7:283–339.

Levy, P. (1948). *Processus stochastiques et mouvement brownien*. Gauthier-Villar, Paris.

MacMillan, L.W. (1986). Analytic Approximation for the American Put Option. *Advances in Futures and Options Research*, 1:119–139.

Margrabe, W. (1978). The Value of an Option to Exchange one Asset for Another. *Journal of Finance*, 33(1):177–186.

Musiela, M., and Rutkowski, M. (1998). *Martingale Methods in Financial Modelling*. Springer-Verlag.

Marchuk, G.I., and Shaidurov, V.V. (1983). *Difference Methods and Their Extrapolations*. Springer-Verlag.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Springer-Verlag.

Perrin, J.B. (1909). *Annales de Chimie et de Physique*, 8me series, September 1909, Translated by F. Soddy, as *Brownian Movement and Molecular Reality*, Taylor and Francis, London, 1910.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition. Cambridge University Press.

Qi, H.D., and Sun, D. (2006). A Quadratically Convergent Newton Method for Computing the Nearest Correlation Matrix. *SIAM Journal on Matrix Analysis and Applications*, 28(2):360–385.

Ramsbottom, J. (1932). Centenary of Robert Brown's Discovery of the Nucleus – Exhibit at Natural History Museum. *The Journal of Botany British and Foreign*, (January):13–16.

Rebonato, R., and Jäckel, P. (1999/2000). The Most General Methodology for Creating a Valid Correlation Matrix for Risk Management and Option Pricing Purposes. *Journal of Risk*, 2(2).

Reiner, E. (1992). Quanto Mechanics. *Risk*, 5(3):59–63.

Roll, R. (1977). An Analytic Valuation Formula for Unprotected American Call Options on Stocks with Known Dividends. *Journal of Econometrics*, 5:251–258.

Shreve, S., Chalasani, P., and Jha, S. (1997). Stochastic Calculus and Finance. Available at: http://www.stat.berkeley.edu/users/evans/shreve.pdf.

Smith, G.D. (1985). *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford University Press.

Stulz, R.M. (1982). Options on the Minimum or Maximum of Two Risky Assets. *Journal of Financial Economics*, 10:161–185.

Tilley, J.A. (1993). Valuing American Options in a Path Simulation Model. *Transactions of the Society of Actuaries*, 45:83–104.

Wiener, N. (1923). Differential Spaces. *Journal of Mathematical Physics*, 2:131–174.

Wiener, N. (1924). Un problème de probabilité dénombrables. *Bulletin de Société Mathématique de France*, 52:569–578.

Whaley, R.E. (1981). On the Valuation of American Call Options on Stocks with Known Dividends. *Journal of Financial Economics*, 9:207–211.

Further reading

Aitchison, J., and Brown, J.A.C. (1966). *The Lognormal Distribution*. Cambridge University Press.

Andersen, L.B.G., and Brotherton-Ratcliffe, R. (1998). The Equity Option Volatility Smile: An Implicit Finite-Difference Approach. *Journal of Computational Finance*, 1(2):5–37.

Anderson, T.W. (1984). *An Introduction to Multivariate Statistical Analysis*, Second Edition. Wiley, New York.

Berndt, E.K., Hall, B.H., Hall, R.E., and Hausman, J.A. (1974). Estimation and Inference in Nonlinear Structural Models. *Annals of Economic and Social Measurement*, 3/4:653–665.

Beyer, W.H. (1982). *CRC Standard Mathematical Tables*. CRC Press, Florida.

Black, F., and Scholes, M. (1973). The Pricing of Corporate Liabilities. *Journal of Political Economy*, 81:637–657.

Bratley, P. (1986). Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators. *ACM Transactions on Mathematical Software*, 12(4):362–376.

Bratley, P., and Fox, B.L. (1988). Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator. *ACM Transactions on Mathematical Software*, 14(1):88–100.

Bratley, P., Fox, B.L., and Niederreiter, H. (1992). Implementation and Tests of Low-Discrepancy Sequences. *ACM Transactions on Modeling and Computer Simulation*, 2(3):195–213.

Brennan, M.J., and Schwartz, E.S. (1978). Finite Difference Methods and Jump Processes Arising in the Pricing of Contingent Claims: A Synthesis. *Journal of Financial and Quantitative Analysis*, 13:462–474.

Chan, T.F., Golub, G.H., and Leveque, R.J. (1982). Updating Formulae and a Pairwise Algorithm for Computing Sample Variances. In: Caussinus, H.,, Tomassone, R.,, Ettinger, P., eds.)*Compstat 1982 Part 1: Proceedings in Computational Statistics*. Physica-Verlag, 1982.

Cotton, I.W. (1975). Remark on Stably Updating Mean and Standard Deviation of Data. *Communications of the ACM*, 18(8):458.

Cox, D.R., and Hinkley, D.V. (1979). *Theoretical Statistics*. Chapman & Hall.

Craig, I.J.D., and Sneyd, A.D. (1988). An Alternating Direction Implicit Scheme for Parabolic Equations with Mixed Derivatives. *Computers & Mathematics with Applications*, 16(4):341–350.

Dickey, J.M. (1967). Multivariate Generalizations of the Multivariate *t* Distribution and the Inverted Multivariate *t* Distribution. *Annals of Mathematical Statistics*, 38(2):511–518.

Duffie, D. (1996). *Dynamic Asset Pricing Theory*, Second Edition. Princeton University Press.

Engle, R.F. (1995). *ARCH: Selected Readings*. Advanced Texts in Econometrics. Oxford University Press.

Faure, H. (1982). Discrépance de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, 41:337–351.

Feller, W. (1971). *An Introduction to Probability Theory and Its Applications, Vol. II*. Wiley.

Freedman, D. (1983). *Brownian Motion and Diffusion*. Springer-Verlag, New York.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York.

Glasserman, P., and Heidelberger, P. (2000). Variance Reduction Techniques for Value-at-Risk with Heavy-Tailed Risk Factors. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A. (eds.), *Proceedings of the 2000 Winter Simulation Conference*.

Goldberger, A.S. (1997). *A Course in Econometrics*. Harvard University Press.

Good, I.J. (1979). Computer Generation of the Exponential Power Distribution. *Journal of Statistical Computation and Simulation*, 9(3):239–240.

Hamilton, J. (1994). *Time Series Analysis*. Princeton University Press.

Hanson, R.J. (1975). Stably Updating Mean and Standard Deviation of Data. *Communications of the ACM*, 18(1):57–58.

Haug, E.G. (1998). *Option Pricing Formulas*. McGraw Hill.

Hunt, P.J., and Kennedy, J.E. (2004). *Financial Derivatives in Theory and Practice*. Wiley.

Jäckel, P. (2002). *Monte Carlo Methods in Finance*. Wiley.

Johnson, N.L., and Kotz, S. (1992). *Distributions in Statistics: Continuous Multivariate Distributions*. Wiley.

Johnson, N.L., Kotz, S., and Kemp, A. (1992). *Univariate Discrete Distributions*. Wiley.

Johnson, N.L., Kotz, S., and Balakvishnam, N. (1994). *Continuous Univariate Distributions*, Second Edition. Wiley.

Johnson, R.A., and Wichern, D.W. (1999). *Applied Multivariate Statistical Analysis*. Prentice Hall.

Jorion, P. (1997). *Value at Risk*. McGraw Hill.
Joshi, M.S. (2004). *The Concepts and Practice of Mathematical Finance*. Cambridge University Press.
Kloeden, P.E., and Platen, E. (1999). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag.
Krzanowski, W.J. (2000). *Principles of Multivariate Analysis: A User's Perspective*. Oxford University Press.
Levy, G. (2004). *Computational Finance: Numerical Methods for Pricing Financial Instruments*. Elsevier.
Mardia, K.V., Kent, J.T., and Bibby, J.M. (1988). *Multivariate Analysis*. Probability and Mathematical Statistics. Academic Press, London.
Martellini, L., and Priaulet, P. (2001). *Fixed-Income Securities: Dynamic Methods for Interest Rate Risk Pricing and Hedging*. John Wiley.
Merton, R.C. (1973). The Theory of Rational Option Pricing. *The Bell Journal of Economy and Management Science*, 4(1):141–181.
Markowitz, H.M. (1989). *Mean-Variance Analysis in Portfolio Choice and Capital Markets*. Blackwell.
McIntyre, R. (1999). Black–Scholes Will Do. *Energy & Power Risk Management*, (November):26–27.
McKee, S., and Mitchell, A.R. (1970). Alternating Direction Methods for Parabolic Equations in Two Space Dimensions with a Mixed Derivative. *The Computer Journal*, 13(1):81–86.
Mitchell, A.R., and Griffiths, D.F. (1980). *The Finite Difference Method in Partial Differential Equations*. Wiley, New York.
Morokoff, W. (1999). The Brownian Bridge E–M Algorithm for Covariance Estimation with Missing Data. *Journal of Computational Finance*, 2(2):75–100.
Morgan, J.P. (1996). *RiskMetrics – Technical Document*, Fourth Edition. New York.
Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM.
Pelsser, A. (2000). *Efficient Methods for Valuing Interest Rate Derivatives*. Springer-Verlag.
Rebonato, R. (1998). *Interest-Rate Option Models*, Second Edition. Wiley.
Richardson, L.F. (1910). The Approximate Arithmetical Solution by Finite Differences od Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam. *Philos. Trans. R. Soc. Lond. A*, 210:307–357.
Richardson, L.F., and Gaunt, G.A. (1927). The Deferred Approach to the Limit. *Philos. Trans. R. Soc. Lond. A*, 226:299–361.
Richardson, L.F. (1927). *Philosophical Transactions of the Royal Society of London, Series A*, 226:299.
Rogers, L.C.G., and Talay, D. (1997). *Numerical Methods in Finance*. Cambridge University Press.
Schonbucher, P.J. (2003). *Credit Derivatives Pricing Models: Model, Pricing and Implementation*. Wiley.
Sobol, I.M. (1967). The Distribution of Points in a Cube and the Approximate Evaluation of Integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112.
Strang, G. (1976). *Linear Algebra and Its Applications*. Academic Press.
Stuart, A., and Ord, J.K. (1987). *Kendall's Advanced Theory of Statistics*, Fifth Edition. Griffin.

West, D.H.D. (1979). Updating Mean and Variance Estimates: An Improved Method. *Communications of the ACM*, 22(9):532–535.

Wilmott, P., Howison, S., and Dewynne, J. (1995). *The Mathematics of Financial Derivatives*. Cambridge University Press.

Index

A

Absolute pricing errors 162
American options 59, 97–179
  call options 99–102
    Black approximation 101, 102
    with cash dividends 97–102
    critical asset values 107–109
    MacMillan–Barone-Adesi–Whaley method 105, 106, 112–114
    pricing errors 135
    Roll–Geske–Whaley approximation 97–101
  implied volatility 81
  put options 179
    critical asset values 109–111
    MacMillan–Barone-Adesi–Whaley method 106, 107, 112–114
    pricing errors 136
  stochastic lattice 172–180
    asset prices 173, 174
    Monte Carlo estimate 177–180
    option prices 174–176
    simulation parameters 173
  two assets 194–200
  vanilla 97–111
    call with cash dividends 97–102
    grid methods 135–167
    lattice methods 114–135
    MacMillan–Barone-Adesi–Whaley method 102–107
    numerical solution of critical asset values 107–111
Amortization 212
Analytic pricing
  down and out call options 86–88
  up and out call options 88–91
Analytics_MathLib function 262–266
Annualized standard deviation 78
Arithmetic progression 323
Asset price, stochastic lattice 173, 174
Asset price index 161
Asset price movements, Brownian motion 9, 10
Asset values 121, 122
Avogadro's number 5

B

Back-substitution 146
Backwards iteration 123–125, 147–150
Barrier options 85–95, 118, 295–302
  down and out call 86–88, 295–298
    Monte Carlo pricing 91–95
  equity 275–280
  foreign exchange 284–288
  up and out call 88–91, 298–301
Base currency 231
Basis swap 212–214
Bayes law 20, 93
BBS *see* binomial Black–Scholes method
BBSR *see* binomial Black–Scholes method, with Richardson extrapolation
BEGKR method 185, 187–189
Binomial Black–Scholes method 131–133
  with Richardson extrapolation 133–135

Binomial lattice 81, 114–122
  backwards iteration 123–125
  with BBS and BBSR 131–135
  computation of Greeks 125–129
  construction and use 120–128
  with control variate 129, 130
  node asset values 121, 122
  terminal node option payoff 122
  values of constants 121
Black approximation 101, 102
black_scholes function 264
Black–Scholes equation 11, 37, 55
  American options 129, 130
  binomial 131–133
    with Richardson extrapolation 133–135
  continuous dividends 72–74
  derivation of 62–65
  discrete dividends 74, 75
  equity quanto options 240–243
  European options 62–83, 129
    multiasset option 65–67
  foreign exchange 229–232
  Greeks 75–77
  historical volatility 78, 79
  implied volatility 79–81
  multiasset options 181, 182
    see also grid methods; MacMillan–Barone-Adesi–Whaley method
Black–Scholes finite-difference approximation 325–328
  general case 325
  log transformation 325–328
  nonuniform grid method 148–155
  uniform grid method 138–142, 160–162, 325–328
Boundary values 142, 143
  lower asset 142
  option maturity 142, 143
  upper asset 142
Box–Muller transformation 42–45, 52
Brownian bridge 19–21
  alternative derivation 329–332
  down and out call options 91–95
  relation to Ornstein–Uhlenbeck bridge 30, 31
Brownian model of asset price movements 9, 10
Brownian motion 5–9, 86
  asset price movements 9, 10
  drift
    changing *see* Girsanov's theorem
    constant 8
    zero 8
  geometric 10, 12, 181
  Ito's quotient rule 17, 18
  multiasset geometric 13–15
  one source of randomness 16, 18
  proofs 333–340
  properties of 6–9
  scaled 22
  time-transformed 21–24
Brown, Robert 5, 7
bs_opt function 82
bs_opt_barrier_downout_call function 88

C
C# code 245, 246
C# portfolio pricing 245–288
  equity deal classes 267–280
    equity barrier option 275–280
    generic equity basket option 270–275
    single equity options 267
    two-equity option 267–269
  foreign exchange deal classes 280–288
    FX barrier option 284–288
    FX forward 280, 281
    single FX option 281–284
  market data file 246
  portfolio configuration file 246
  portfolio definition file 246
    broad-investments 250–254
    EQ-investments 249, 250, 253, 254
  portfolio driver file 249

portfolio valuer application 248, 249
PricingUtils and Analytics_MathLib 262–266
storing/retrieving market data 254–262
CurrencyTable 255–259
EquityTable and CorrelationTable 259–262
Call options
American 99–102
Black approximation 101, 102
with cash dividends 97–102
critical asset values 107–109
MacMillan–Barone-Adesi–Whaley method 105, 106, 112–114
pricing errors 135
Roll–Geske–Whaley approximation 97–101
double knockout 166–171
down and in 85
down and out 85–88, 168
analytic pricing 86–88
Brownian bridge 91–95
Monte Carlo pricing 91–95
nonuniform grid method 153, 154
European 60, 61, 64, 65, 69, 73, 74, 76, 77, 79, 80, 82
multiasset
four assets 208
three assets 201, 204, 205
two assets 197, 198
up and in 85
up and out 85, 88–91
analytic pricing 88–91
vanilla
American 97–111
European 59, 69, 83, 84
Caplet, quanto 223–225
Central limit theorem 303, 304
Cholesky decomposition 48, 50, 52
Closed form solutions 181
Conditional mean 310, 311
Constant drift 8
Continuous dividends 61, 62, 72–74
Continuous hazard rate 233
Continuously compounded spot rate 209
Control variate technique 129, 130
Correlated variates 47–58
correlation and covariance 46, 47
lognormal distribution 56–58
normal distribution 51–55
repairing correlation and covariance matrices 48–51
Correlation 47
Correlation matrix 13, 47
optimally repaired 49–51
repair of 48–51
CorrelationTable 259–262
Coupon payment 211
early 217
floating leg 212
late 217, 218
on time 215, 216
Covariance 307–309
estimation of 47
four variables 308
normal distribution 310, 311
n variables 308, 309
three variables 307, 308
two variables 307
unconditional 23, 24
Covariance matrix 182, 309
repair of 48–51
Covered interest arbitrage 228, 229
Cox–Rubinstein–Ross binomial model 188
Crank–Nicolson method 141, 159, 162, 163
Credit default swap 235, 236
Credit derivatives 232–237
credit default swap 235, 236
defaultable bond 235
hazard rate 232, 233
continuous 233

estimation from market observables 234, 235
total return swap 236, 237
Credit risk 2
Critical asset values, numerical solution 107–111
CRR lattice 113
Cumulative normal distribution function 322, 323
CurrencyTable 255–259
Current value 59

D
Defaultable bond 235
Delta 75, 119, 169
computation of 125
vanilla European options 291, 292
Depth first procedure 177
Differential swaplet *see* quanto, swaplet
Diff swaplet *see* quanto, swaplet
Discrete dividends 60, 61, 74, 75
Dividends
continuous 61, 62, 72–74
discrete 60, 61
Double knockout call option 166–171
Greeks 169
Down and in call options 85
Down and out call options 85–88, 168
analytic pricing 86–88
Brownian bridge 91–95
derivation of 295–298
Monte Carlo pricing 91–95
nonuniform grid method 153, 154
DownOutEquityOption-Deal 279
Drunkard's walk *see* Brownian motion

E
Early coupon payment 217
Early exercise 147–150
Eigen decomposition 48
Equity barrier option 275–280
Equity deal classes 267–280
equity barrier option 275–280
generic equity basket option 270–275
single equity options 267
two-equity option 267–269
Equity derivatives 237–243
quantos 240–243
equity quanto forward 242, 243
quanto adjustment factor 241, 242
total return swap 237–240
equity leg 237, 238
equity swap 239, 240
floating leg 238
payer equity 238, 239
EquityOptionDeal 265–267
Equity quanto forward 242, 243
EquityTable 259–262
European options 59–95
barrier options 85–95
down and out call options 86–88
Monte Carlo pricing of down and out options 91–95
up and out call options 88–91
call options 60, 61, 64, 65, 69, 73, 74, 76, 77, 79, 80, 82
double knockout 166, 167, 169–171
down and in 85
down and out 85–88, 155, 156, 168
two assets 197, 198
foreign exchange 229–232
implied volatility 79–81
martingale measure 59, 60
multiasset
four assets 207, 208
three assets 201, 204, 205
two assets 190–194
put call parity 60–62
continuous dividends 61, 62
discrete dividends 60, 61
put options 59
four assets 208
three assets 202, 204, 205

two assets 197, 198
vanilla 59, 62–83, 85
Black–Scholes equation 62–65
call options 59–85
Greeks for 289–294
put options 59
volatility
historical 78, 79
implied 79–81
Exotic options 83, 97, 118, 129, 147, 168, 183
Explicit method 141

F
Faure sequence 41
Feynman–Kac formula 64, 341, 342
Filtration 6
Financial derivatives 1–3
Finite-difference approximation
see Black–Scholes finite-difference approximation
Floorlet, quanto 226
Foreign exchange call 231, 232
Foreign exchange deal classes 280–288
FX barrier option 284–288
FX forward 280, 281
single FX option 281–284
Foreign exchange derivatives 228–232
covered interest arbitrage 228, 229
European option 229–232
FX forward 229
Foreign exchange forward 1, 229
Foreign exchange option 2, 3
Forward rate agreement 210
Four asset options 205–208
FourEquityOptionDeal 273
Fubini's theorem 26, 31
Fully implicit method 141
FX *see* foreign exchange
FX forward 229, 280, 281

G
Gamma 75, 119, 169
computation of 125
vanilla European options 290, 291
Gamma function 321, 322
Gaussian distribution *see* normal distribution
General error distribution 319, 320
kurtosis 319, 320
shape parameter a 320
value of λ for variance h_i 319
generic equity basket option 270–275
GenericEquityBasketOptionDeal 270–275
Geometric Brownian motion 10, 12, 181
Geometric progression 323
Girsanov's theorem 12, 13, 68, 72
Going short 63, 66
Greeks 119
binomial lattice 125–129
Black–Scholes equation 75–77
double knockout call option 169
vanilla European options 289–294
see also individual Greeks
Grid methods 135–167
double knockout call option 166–169
log transformation
nonuniform grids 162–165
uniform grids 156–163
nonuniform grids 148–159
down and out call option 153, 154
finite-difference approximation 149–155
log transformation 162–165
uniform grids 136–150
backwards iteration and early exercise 147–150
boundary conditions 142, 143
finite-difference approximation 138–142
log transformation 156–163
option values 143–147

H
Hazard rate 232, 233

continuous 233
estimation from market observables 234, 235
Hazard rate curve 233
Heavy tail distributions 183
Hedge statistics *see* Greeks

I
ICurve 257–259
Implicit method 141
implied_volatility function 81
Implied volatility
American options 81
European options 79–81
Independent variates 41–46
lognormal distribution 45, 46
normal distribution 42–44
Student's t-distribution 46
Integrals
barrier option 295–302
down and out call 295–298
up and out call 298–302
standard 321
stochastic 33
Interest rate derivatives 209–227
continuously compounded spot rate 209, 210
forward rate agreement 210
quantos 223–227
caplet 223–225
floorlet 226
swaplet 227
simply compounded spot rate 210
timing adjustment 218–223
Interest rate swap 211–218
amortization 212
basis swap 212–214
coupon payment 215–218
floating leg 212
general payment timing 216, 217
swap rate 212
vanilla 211
Ito's formula 10–12
multiasset geometric Brownian motion 13–15
two-dimensional 67
Ito's isometry 26, 32
Ito's product rule 15, 16
n dimensions 18, 19
Ito's quotient rule 16–18, 241

K
Knockin options 83
Knockout options 83, 92

L
Late coupon payment 217, 218
Lattice methods
binomial lattice 81, 114–122
backwards iteration 123–125
with BBS and BBSR 131–135
computation of Greeks 125–129
construction and use 120–128
with control variate 129, 130
node asset values 121, 122
terminal node option payoff 122
values of constants 121
multiasset options 185–189
stochastic lattice 172–180
asset prices 173, 174
Monte Carlo estimate 177–180
option prices 174–176
simulation parameters 173
Law of large numbers 303
Lockout periods 118
Lognormal distribution 45, 46, 55–58, 114, 315, 316
Lognormal mean 114
Lognormal variance 115
log transformation
nonuniform grids 162–165
uniform grids 160–163
London Inter Bank Offer Rate (LIBOR) 213
Low discrepancy sequences 38

M
MacMillan–Barone-Adesi–Whaley method 102–107, 112–114
see also Black–Scholes equation

Main currency 214
MarketDataDictionaries 254, 255, 259–262
Market data file 246
Market data, storing/retrieving 254–262
  CurrencyTable 255–259
  EquityTable and CorrelationTable 259–262
Market observables 233–235
Market rate dictionaries 250
Markov process 6
Martingale measure 6, 59, 60
Maturation 59
Mean 25
  Ornstein–Uhlenbeck process 25
  unconditional 23
Microsoft Excel
  CALCULATE OPTIONS 83
  NORMDIST 83
  pricing options 82, 83
Moment generating functions 311, 312
Monte Carlo simulation 37, 172
  American option 177–180
  down and out options 91–95
  multiasset options 183–185
  with random numbers 40, 41
Multiasset geometric Brownian motion 13–15
Multiasset options 181–208
  Black–Scholes equation 65–67, 181, 182
  four assets 205–208
  lattices 185–189
  Monte Carlo methods 183–185
  three assets 201–205
  two assets 190–201
    American options 194–200
    European exchange options 190–192
    European options on maximum or minimum 192–196
Multivariate distributions 41–46
  generation of 47–58
  lognormal distribution 45, 46 55–58, 114, 315–316
  normal distribution *see* normal distribution
  Student's *t* distribution 46, 317–318

N
Neiderreiter sequence 39, 40
Newton's method 79, 80, 81, 98
Nonuniform grids 150–159
  down and out call option 153, 154
  finite-difference approximation 149–155
  log transformation 162–165
Normal distribution 11, 21, 42–44, 51–55, 313–315
  conditional mean 310, 311
  covariance 310, 311
  cumulative function 322, 323
  mean 314
  variance 314, 315
Numeraire 60, 218–222

O
Obligation 1
Option payoff 122
Option prices 174–176
Option values 143–147
Ornstein–Uhlenbeck bridge 27–31
  relation to Brownian bridge 30, 31
Ornstein–Uhlenbeck process 22–27
  mean 25
  unconditional mean 23, 25
  unconditional variance/covariance 23, 24
  variance 25, 26

P
Payer equity total return swap 238, 239
Payer interest rate swap 211
Payment timing 216, 217
Payoff 59
Portfolio configuration file 246
Portfolio definition file 246

broad-investments 250–253, 254
EQ-investments 249, 250, 253, 254
Portfolio driver file 249
Present value 60
Pricing errors 135, 136
Pricing options, Microsoft Excel 82, 83
PricingUtils 262–266
Principal exchange 214
Pseudo-random sequences 38–41
Put call parity 77
continuous dividends 61, 62
discrete dividends 60, 61
Put options
American 179
critical asset values 109–111
MacMillan–Barone-Adesi–Whaley method 106, 107, 112–114
pricing errors 136
European 59
four assets 208
three assets 202, 204, 205
two assets 197, 198
multiasset
four assets 208
three assets 201, 204
two assets 194, 197, 198

Q

Quantos
equity 240–243
equity quanto forward 242, 243
quanto adjustment factor 241, 242
interest rate 223–227
caplet 223–225
floorlet 226
swaplet 227
Quasirandom_Normal_LogNormal_Correlated function 53–55
Quasi-random sequences 38–41

R

Radon–Nikodym derivative 12
Random variates 37–58
correlated variates 47–58
independent 41–46
lognormal distribution 45, 46
normal distribution 42–44
Student's *t*-distribution 46
pseudo-random/quasi-random sequences 38–41
Random walk *see* Brownian motion
Rate swap 211–218
amortization 212
basis swap 212–214
coupon payment
early 217
late 217, 218
on time 215, 216
floating leg 212
general payment timing 216, 217
payer 211
receiver 211
swap rate 212
vanilla 211
Receiver interest rate swap 211
Return 9
Rho 75
vanilla European options 293
RiskFreeRate 257
Roll–Geske–Whaley approximation 97–101

S

Scaled Brownian motion 22
Siedentopf, Henry 5
Simply compounded spot rate 210
Single equity options 267
Single FX option 281–284
Sobol sequences 39, 41, 184, 185
Spot rate
continuously compounded 209
simply compounded 210
Standard deviation, annualized 78
Standard integrals 321
Stochastic integral, expectation of 33
Stochastic lattice 172–180
asset prices 173–174

Monte Carlo estimate 177–180
option prices 174–176
simulation parameters 173
Stochastic processes 5–35
Brownian bridge 19–21
Brownian model of asset price movements 9, 10
Brownian motion 5–9
time-transformed 21–24
Girsanov's theorem 12, 13
Ito's product
in *n* dimensions 18, 19
and quotient rules 15–18
Ito's formula 10–12
multiasset geometric Brownian motion 13–15
Ornstein–Uhlenbeck bridge 27–31
Ornstein–Uhlenbeck process 22–27
Strike price 59
Structured deal 239
Student's *t*-distribution 46, 317, 318
Swaplets 211
quanto 227
Swap rate 212

T
Taylor expansion 10, 14, 151
Theta 75, 119, 169
computation of 125, 126
vanilla European options 292, 293
Three asset options 201–205
Time-transformed Brownian motion 21–24
Timing adjustment 218–223
Total return swap
credit 236, 237
equity 237–240
equity leg 237, 238
equity swap 239, 240
floating leg 238
payer equity 238, 239
Trading days 78
Two asset options 190–201
American 197–200
European 190–192
maximum or minimum 192–194
Two-equity option 267–269

U
Unconditional mean 23
Unconditional variance 23, 24
Uniform grids 136–150
backwards iteration and early exercise 147–150
boundary conditions 142, 143
finite-difference approximation 138–142
log transformation 156–163
option values 143–147
Up and in call options 86
Up and out call options 83, 88–91
analytic pricing 88–91
derivation of 298–302

V
Vanilla options
American 97–111
call with cash dividends 97–102
grid methods 135–167
lattice methods 114–135
MacMillan–Barone-Adesi–Whaley method 102–107
numerical solution of critical asset values 107–111
binomial lattice 81, 114–122
with BBS and BBSR 131–135
construction and use 120–128
with control variate 129, 130
European 59, 62–84
call 59, 69, 83, 85
Greeks for 289–294
put 59
grid methods 135–167
double knockout call option 166–169
nonuniform grids 148–159
uniform grids 136–150
interest rate swap 211
Variance 25, 26, 305–307
n variables 306, 307

one variable 305
Ornstein–Uhlenbeck process 25, 26
three variables 306
two variables 305
unconditional 23, 24
Vega 76
computation of 126
vanilla European options 294
Visual Basic 82–84
bs_opt 84
bs_opt_barrier_downout_call 88
Volatility 7
historical 78, 79
implied 79–81
volatility smile 79

W
Wiener, Norbert 5
Wiener process *see* Brownian motion

Y
YieldCurve 257

Z
Zero coupon rate 257
Zero drift 8
Zsigmondy, Richard 5

Glossary

The notation used is as follows:

GBM	Geometric Brownian motion
BM	Brownian motion
W_t	Brownian motion at time t
ρ	the correlation coefficient
$E[x]$	the expectation value of X
$\mathrm{Var}[X]$	the variance of X
$\mathrm{Cov}[X, Y]$	the covariance between X and Y
$\mathrm{Cov}[X]$	the covariance between the variates contained in the vector X
σ	the volatility. Since assets are assumed to follow GBM it is computed as the *annualized* standard deviation of the n continuously compounded returns
$N_1(a)$	the univariate cumulative normal distribution function. It gives the cumulative probability, in a standardized univariate normal distribution, that the variable x_1 satisfied $x_1 \leqslant a$
$N_2(a, b, \rho)$	the bivariate cumulative normal distribution. It gives the cumulative probability, in a standardized bivariate normal distribution, that the variables x_1 and x_2 satisfy $x_1 \leqslant a$ and $x_2 \leqslant b$ when with correlation coefficient between x_1 and x_2 is ρ
r	the risk free interest rate
q	the continously compounded dividend yield
S_{it}	the ith asset price at time t
I_{nn}	the n by n unit matrix
$\Lambda(\mu, \sigma^2)$	a lognormal distribution with parameters μ and σ^2. If $y = \log(x)$ and $y \sim \mathrm{N}(\mu, \sigma^2)$ then the distribution for $x = \mathrm{e}^y$ is $x \sim \Lambda(\mu, \sigma^2)$. We have $E[x] = \exp(\mu + \frac{\sigma^2}{2})$ and $\mathrm{Var}[x] = \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$

Continued on back cover

Glossary (*Continued*)

$DF(t, T)$ the discount factor between times t and T, where $T \geqslant t$. The price of a nondefaultable zero coupon bond which matures at time T is the expected value of $DF(t, T)$. In this book we assume that interest rates are deterministic and thus $DF(t, T)$ is the value of a nondefaultable zero coupon bond maturing at T

$\overline{DF}(t, T)$ the discount factor (including the possibility of default) between times t and T, where $T \geqslant t$: $\overline{DF}(t, T) = S(t, T)DF(t, T)$. The price of a defaultable zero coupon bond which matures at time T is the expected value of $\overline{DF}(t, T)$. In this book we assume that interest rates are deterministic and thus $\overline{DF}(t, T)$ is the value of a defaultable zero coupon bond maturing at T

$F(t, T_1, T_2)$ the forward rate at time t between times T_1 and T_2 where $T_2 \geqslant T_1$ and $T_1 \geqslant t$

$L(T_1, T_2)$ the simply compounded spot rate between times T_1 and T_2, where $T_2 \geqslant T_1$

$\log(x)$ the natural logarithm of x

$\mathrm{N}(a, b)$ normal distribution, with mean a and variance b

$\mathrm{d}W_t$ a normal variate (sampled at time t) from the distribution $\mathrm{N}(0, \mathrm{d}t)$, where $\mathrm{d}t$ a specified time interval e.g. $\mathrm{d}x = \mu\,\mathrm{d}t + \mathrm{d}W_t$

$\mathrm{d}Z_t$ a normal variate (sampled at time t) from the distribution $\mathrm{N}(0, 1)$. Note: The variate $\mathrm{d}\psi = \sqrt{\mathrm{d}t}\,\mathrm{d}Z_t$ has the same distribution as $\mathrm{d}W_t$

IID independently and identically distributed

$\mathrm{U}(a, b)$ the uniform distribution, with lower limit a and upper limit b

$|x|$ the absolute value of the variable x

PDF the probability density function of a given distribution

$x \wedge y$ the minimum of x and y, that is $\min(x, y)$

$S(t, T)$ the survival probability between time t and T, $T > t$

$\|A - B\|$ the distance between two matrices with the same dimensions. If A and B both have n rows and m columns then this distance is:

$$\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\{A_{i,j} - B_{i,j}\}^2}$$

where $A_{i,j}$ and $B_{i,j}$ refer to the element in the ith row and jth column